المملكـة العربيــة الســعودية

المؤسسة العامة للتعليم الفني والتدريب المهني

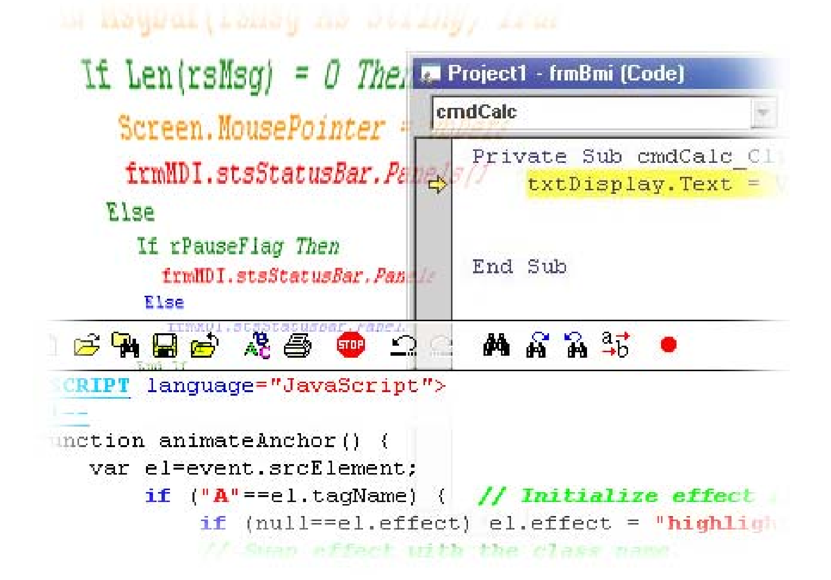
الإدارة العامة لتصميم وتطوير المناهج



برمجيات

تصميم قواعد البيانات

١٦٢ حاب



مقدمة

الحمد لله وحده، والصلاة والسلام على من لا نبي بعده، محمد وعلى آله وصحبه، وبعد:

تسعى المؤسسة العامة للتعليم الفني والتدريب المهني لتأهيل الكوادر الوطنية المدربة القادرة على شغل الوظائف التقنية والفنية والمهنية المتوفرة في سوق العمل، ويأتي هذا الاهتمام نتيجة للتوجهات السديدة من لدن قادة هذا الوطن التي تصب في مجملها نحو إيجاد وطن متكامل يعتمد ذاتياً على موارده وعلى قوة شبابه المسلح بالعلم والإيمان من أجل الاستمرار قدماً في دفع عجلة التقدم التنموي، لتصل بعون الله تعالى لمصاف الدول المتقدمة صناعياً.

وقد خطت الإدارة العامة لتصميم وتطوير المناهج خطوة إيجابية تتفق مع التجارب الدولية المتقدمة في بناء البرامج التدريبية، وفق أساليب علمية حديثة تحاكي متطلبات سوق العمل بكافة تخصصاته لتلبي متطلباته، وقد تمثلت هذه الخطوة في مشروع إعداد المعايير المهنية الوطنية الذي يمثل الركيزة الأساسية في بناء البرامج التدريبية، إذ تعتمد المعايير في بنائها على تشكيل لجان تخصصية تمثل سوق العمل والمؤسسة العامة للتعليم الفني والتدريب المهني بحيث تتوافق الرؤية العلمية مع الواقع العملي الذي تفرضه متطلبات سوق العمل، لتخرج هذه اللجان في النهاية بنظرة متكاملة لبرنامج تدريبي أكثر التصاقاً بسوق العمل، وأكثر واقعية في تحقيق متطلباته الأساسية.

وتتناول هذه الحقيبة التدريبية " تصميم قواعد البيانات" لمتدربي قسم" برمجيات " للكليات التقنية موضوعات حيوية تتناول كيفية اكتساب المهارات اللازمة لهذا التخصص.

والإدارة العامة لتصميم وتطوير المناهج وهي تضع بين يديك هذه الحقيبة التدريبية تأمل من الله عز وجل أن تسهم بشكل مباشر في تأصيل المهارات الضرورية اللازمة، بأسلوب مبسط يخلو من التعقيد، وبالاستعانة بالتطبيقات والأشكال التي تدعم عملية اكتساب هذه المهارات.

والله نسأل أن يوفق القائمين على إعدادها والمستفيدين منها لما يحبه ويرضاه، إنه سميع مجيب الدعاء.

الإدارة العامة لتصميم وتطوير المناهج

المملكـة العربيــة الســعودية
المؤسسة العامة للتعليم الفني والتدريب المهني
الإدارة العامة لتصميم وتطوير المناهج



تصميم قواعد البيانات

مقدمة لتصميم قواعد البيانات

مقدمة لتصميم قواعد البيانات

١

الجدارة:

القدرة على وصف مكونات نظام قاعدة البيانات ودورة حياة النظام

الأهداف:

١. أن يتعرف المتدرب مكونات نظام قاعدة البيانات
٢. أن يتعرف المتدرب دورة الحياة لنظام قاعدة البيانات

مستوى الأداء المطلوب:

أن يتقن المتدرب وصف مكونات النظام ومراحل تطويره بنسبة ١٠٠٪.

الوقت المتوقع للتدريب:

ساعتان

الوسائل المساعدة:

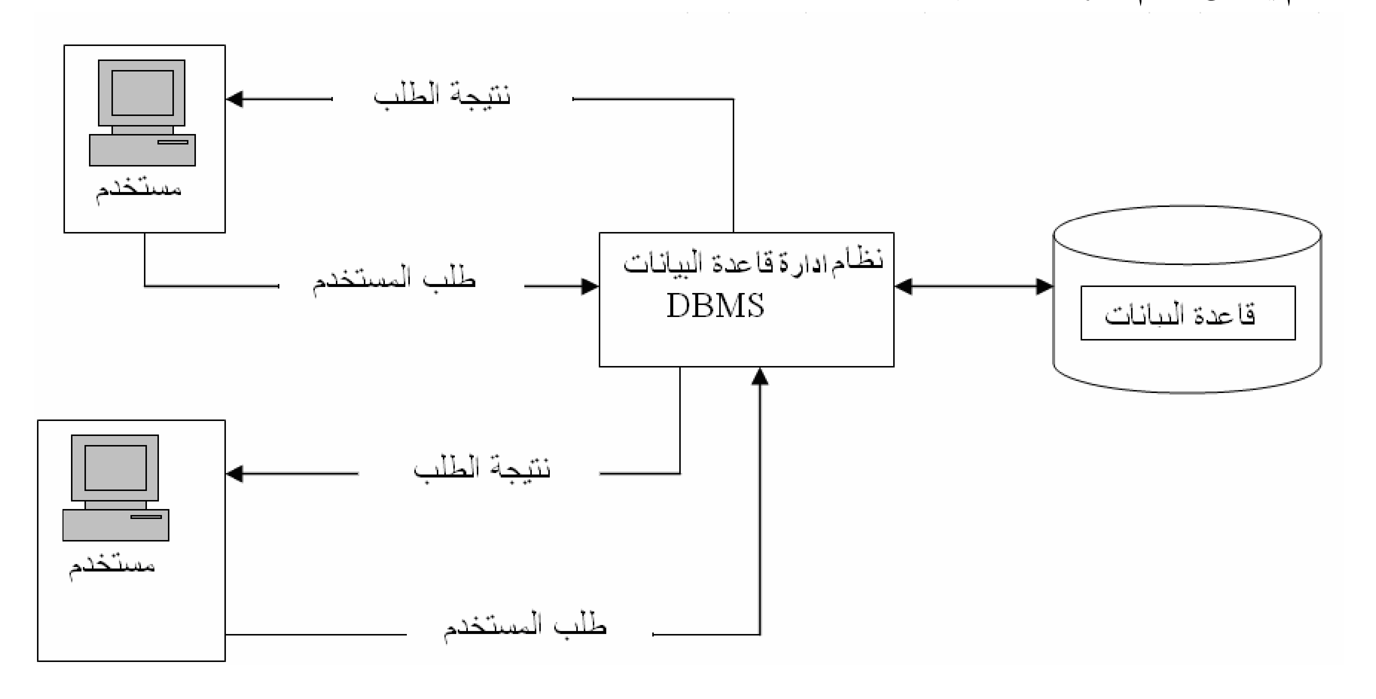
قلم + دفتر

متطلبات الجدارة:

أن يكون المتدرب قد أتقن جميع الجدارت في مقدمة قواعد البيانات .

تعريف قاعدة البيانات

قاعدة البيانات: هي عبارة عن مجموعة المعلومات والبيانات المخزنة بطريقة نموذجية ودون تكرار والمتصلة مع بعضها وفق علاقات متبادلة..ومن أمثلة قواعد البيانات نظام تسجيل المتدربين حيث يقوم على تخزين البيانات الخاصة بالمتدربين والمتدربين والمقررات والشعب... الخ في جداول. وكذلك تحديد العلاقات بين هذه الجداول وفق أسس محددة وثابتة تعتمد على قواعد العمل في هذا النظام وكذلك على استخدام الطرق الصحيحة في عملية تصميم قاعدة البيانات. وتكون قاعدة البيانات مفصولة عن البرامج والتطبيقات التي تقوم بمعالجة هذه البيانات مثل برامج الإدخال والتعديل والحذف ويدير قاعدة البيانات نظام يمسى نظام إدارة قاعدة البيانات.



تعريف نظام إدارة قاعدة البيانات

ما هي إدارة نظام قاعدة البيانات DBMS (Database Management Information System) ؟

هي عبارة عن مجموعة البرامج التي تدير وتتحكم بعملية تخزين واسترجاع البيانات، وتوفر كذلك إمكانية قيام عدد كبير من المستخدمين من الوصول والتعامل مع البيانات، وينظر إليها كذلك على أنها حلقة الوصل بين المستخدمين وقاعدة البيانات، بحيث تقوم باستقبال متطلبات المستخدمين ومن ثم نقلها إلى قاعدة البيانات وتنفيذ البرامج اللازمة لتنفيذ هذه المتطلبات ومن ثم تزويد المستخدم بالنتائج المطلوبة.

مكونات نظام قاعدة البيانات

يقسم نظام قاعدة البيانات إلى خمسة أقسام :

١ - **المكونات المادية (Hardware) :**

وتشمل جميع الأجهزة المادية في النظام مثل الحاسبات، الأجهزة الطرفية، الطابعات وكذلك أجهزة الاتصال في بيئة قاعد البيانات الموسعة... الخ.

٢ - **البرمجيات (Software):**

وهي مجموعة البرامج المستخدمة في قاعدة البيانات، وتقسم إلى ثلاثة أقسام:

أ - **أنظمة التشغيل**: وهي البرامج التي تقوم بإدارة الأجهزة وتهيئتها للعمل وتمكين بقية البرامج من العمل مثل ....Linux, Unix, Windows.

ب - **برنامج قاعدة البيانات**: وهو البرنامج الذي يتولى إدارة قاعدة البيانات مثل Oracle, Sybase, DB2 ... .

ج - **البرامج التطبيقية والبرامج المساعدة**: وهي البرامج التي تقوم بعمليات الاسترجاع والتخزين وكذلك استخراج التقارير... .

٣ - **المستخدمون** : وهم عبارة عن الأشخاص الذين يقومون بالعمل في بيئة قاعدة البيانات وهم :

أ - مدير **النظام**: وهو الشخص المسؤول عن إدارة عمل البيئة العامة التي يعمل بها نظام قاعدة البيانات ويقوم بما يلي:

١. بإدارة المستخدمين ومنح الصلاحيات لاستخدام النظام.
٢. إدارة أجهزة التخزين والأجهزة الأخرى.
٣. متابعة عمل النظام.

ب -**مدير قاعدة البيانات**: وهو المسؤول عن إدارة قاعدة البيانات وتشمل واجباته:

١. تحديد متطلبات قاعدة البيانات من برامج وتجهيزات.
٢. متابعة نظام قاعدة البيانات وتنسيق عملية استخدامه.
٣. توفير الأمن والحماية للنظام.
٤. تصميم آليات المحافظة على قاعدة البيانات وتحديد الإجراءات اللازمة لتوفير الخدمات للمستخدمين الآخرين.

ج - **مصمم قاعدة البيانات** وهو الشخص(الأشخاص) الذي يقوم بعملية تصميم قاعدة البيانات وتشمل واجباته:

١. تحديد البيانات الواجب تخزينها في قاعدة البيانات
٢. تصميم أفضل التراكيب لحفظ البيانات .
٣. تصميم قاعدة بيانات خالية من التكرار .
٤. تحديد طرق الوصول والمعالجة والاسترجاع للبيانات من خلال تصميم الشاشات والتقارير الواجب استخدامها .
٥. توثيق عملية التصميم وطرق الوصول للبيانات .

د - **المبرمجون ومحللو النظم** : وهم الأشخاص الذين يقومون بعملية تصميم البرامج وتنفيذها وتشمل واجباتهم :

١. تصميم التطبيقات وتحويلها إلى برامج بلغة(لغات) برمجة حسب السياسات المقررة في عملية التصميم .
٢. تنفيذ وتطبيق تلك البرامج والتأكد من سلامتها .
٣. عمل الصيانة اللازمة لتلك البرامج .

هـ - **المستخدم النهائي**: وهو الشخص أو مجموعة الأشخاص الذين يقومون بالعمل اليومي على النظام وتطبيق البرامج في مجال محدد مثل الاسترجاع، التعديل، الحذف، تنفيذ التقارير... الخ .

٤ - **الإجراءات والعمليات**: وهي عبارة عن القوانين والتعليمات التي تحكم عمل قاعدة البيانات بشكل صحيح وتكون على شكل تعليمات موثقة بشكل واضح ومحدد.

٥ - **البيانات**: وهي أهم مكونات النظام حيث تشمل مجموعة الحقائق المخزنة في قاعدة البيانات. وكون البيانات تكون على شكل بدائي إذ لابد من تحديد مكان وكيفية التخزين لهذه البيانات حتى تسهل عملية معالجتها والاستفادة منها وهذا عمل المصمم .

أهمية تصميم قواعد البيانات :

إن عملية بناء قاعدة بيانات جيدة لا يأتي بتلك السهولة، إذ لابد من بذل جهد كبير للحصول على قاعدة بيانات جيدة. والتصميم الجيد لقاعدة البيانات يسهل عملية استخدام وإدارة هذه القاعدة أما التصميم السيئ فسيؤدي إلى تكرار البيانات(ويعني وجود نفس البيانات في أكثر من مكان) وبالتالي تصعب عملية الحفاظ على توافقية البيانات وعادة ما يؤدي تكرار البيانات إلى نتائج غير صحيحة عند طلب تلك البيانات من تلك القاعدة وهذا بدوره يؤدي إلى أن أي قرارات إدارية وكذلك أي تخطيط مستقبلي سيكون خاطئا لاعتماده على معلومات غير صحيحة.

دورة الحياة لنظام قاعدة البيانات :

١ - **الدراسة المبدئية للنظام القائم وتشمل ما يلي:**

أ - تحليل الوضع الحالي للمؤسسة ومعرفة طبيعة الإجراءات المستخدمة والتعليمات وقواعد العمل .

ب -تحديد المشاكل التي تواجه النظام المستخدم وكذلك القيود المادية مثل الطاقة البشرية والتمويل المتوفر لتطوير أو استبدال النظام الحالي .

ج - تحديد الأهداف الواجب تحقيقها والمزايا المطلوبة في النظام الجديد.

٢ - **تصميم قاعدة البيانات**: وتعتبر هذه المرحلة من أهم المراحل في دورة حياة النظام إذ لابد من بذل جهد كبير لتصميم النظام للوصول إلى نظام جيد وتؤدي الأهداف المرجوة من عمل النظام وتشمل عملية التصميم ما يلي:

أ - بناء نموذج المفاهيم وتشمل هذه العملية عدة خطوات(سنتطرق إلى هذه العملية بالتفصيل في الفصول اللاحقة):

١. تحليل البيانات ومتطلبات المستخدمين والإجراءات المطلوبة
٢. تعريف وتحديد الكيانات وخصائصها وعلاقتها مع بعضها وكذلك وضعها في الصيغة المعيارية.
٣. رسم مخطط المفاهيم وهو عبارة عن نموذج رسومي يوصف كيانات النظام وعلاقتها مع بعضها.
٤. تعديل النموذج بحيث يشمل الإجراءات الرئيسة، وقواعد عمليات الإضافة والتعديل والحذف على البيانات والتقارير، والشاشات، ومقدار التشاركية و توافقية البيانات....

.

ب - اختيار نظام إدارة قاعدة البيانات (DBMS).

ج - تحويل نموذج المفاهيم إلى نموذج داخلي بالاعتماد على نظام إدارة قاعدة البيانات(DBMS).

د - التصميم المادي وتتم خلاله عملية وضع مواصفات التخزين والوسائط المستخدمة في عملية التخزين وطرق الوصول للبيانات بالاعتماد على نظام إدارة قاعدة البيانات (DBMS).

٣ - **تنفيذ النظام:** وخلال هذه المرحلة تتم عملية إنشاء الجداول وكتابة جميع البرامج اللازمة لتنفيذ متطلبات النظام من الشاشات المختلفة و التقارير المطلوبة ... .

٤ - **عملية الفحص والتقييم للنظام وتشمل :**

أ - فحص قاعدة البيانات والتأكد من عملها بشكل صحيح.

ب - تقييم عمل البرامج والتطبيقات المستخدمة.

٥ - **تطبيق النظام في مكان العمل:** وتشمل هذه العملية عمليات إنشاء الجداول والمستخدمين والصلاحيات..، وتحميل جميع البرامج والتطبيقات وتنفيذها في البيئة الحقيقة التي يجب أن يعمل بها النظام.

٦ -**متابعة عمل النظام:** وهذه العملية تستمر طيلة فترة حياة النظام للتأكد من عمله بشكل صحيح وكذلك تعديل النظام ليتواءم مع المتطلبات الجديدة لبيئة العمل مثل تغير القوانين والأنظمة وقواعد العمل.

تمارين

١ - أي العبارات التالية صحيح وأيها خاطئة ؟

أ - من واجبات المبرمج توفير الأمن والحماية للنظام

ب -المستخدم النهائي هو الشخص الذي يقوم بتوثيق عملية التصميم وطرق الوصول للبيانات.

ج - تعتبر البيانات من أهم مكونات نظام قاعدة البيانات.

٢ - عرف ما يلي:

- قاعدة البيانات
- نظام إدارة قاعدة البيانات

٣ - ما هي أهمية تصميم قاعدة البيانات ؟

٤ - اشرح مرحلة تصميم قاعدة البيانات شرحا مفصلا .

٥ - اذكر واجبات كل من مصمم قاعدة البيانات، المبرمجين ومحللي النظم.



تصميم قواعد البيانات

قواعد البيانات العلائقية

قواعد البيانات العلائقية

٢

الجدارة:

القدرة على وصف قواعد البيانات العلائقية وصفا صحيحا .

الأهداف:

- أن يتعرف المتدرب على قاعدة البيانات العلائقية .
- أن يميز المتدرب بين مختلف أنواع المفاتيح للجداول (العلاقة).
- أن يتعرف المتدرب على مختلف أنواع التشاركية بين الجداول (العلاقات).

مستوى الأداء المطلوب:

أن يصف المتدرب قواعد البيانات العلائقية وصفا صحيحا وكاملا بنسبة ١٠٠ ٪.

الوقت المتوقع للتدريب:

ساعتان.

الوسائل المساعدة:

قلم + دفتر.

متطلبات الجدارة:

أن يكون المتدرب قد أتقن الجدارة في الوحدة السابقة.

قاعدة البيانات العلائقية:

بدأ نشوء مفهوم قواعد البيانات العلائقية عام ١٩٧٠ عندما قدم العالم Codd اقتراحا لهذا النموذج والذي تم بناؤه على **نظريات الجبر العلائقي** ومن هنا برزت قوة هذا النموذج وسرعة انتشاره فيما بعد. ففي مطلع الثمانينات بدأت الكثير من الشركات بتبني هذا النموذج وتطبقيه، فنلاحظ الآن أن معظم أنظمة قواعد البيانات الموجودة في الأسواق تتوافق مع هذا النموذج. وتتلخص فكرة النموذج في النظر إلى قاعدة البيانات على أنها مجموعة من الجداول (Tables) أو علاقات تسمى (Relations) ومن هنا جاءت تسمية النموذج وكل جدول يجب أن يكون له اسم (لا يوجد أكثر من جدول يحمل نفس الاسم).والعلاقة هي عبارة عن مصطلح رياضي وتمثل جدولاً ذا بعدين (صفوف وأعمدة)، ولا توجد هناك أهمية لترتيب الصفوف أو الأعمدة.حيث تمثل الصفوف مجموعة سجلات الجدول(Records or Tuple) وتمثل الأعمدة الصفات لهذا الجدول (Attributes) ويجب أن يكون لكل صفة مجال (Domain) من القيم التي يمكن أن يحتويها هذا العامود. وترتبط هذه الجداول مع بعضها بواسطة روابط..ويجب أن يكون لكل جدول مفتاح رئيس (Primary Key) لتمييز الصفوف عن بعضها والنقطة التي تمثل تقاطع الصف مع العمود (الصفة) تمثل قيمة لهذا الصف.و سنقوم في بقية أجزاء هذه الوحدة بتقديم وصفاً لقواعد البيانات العلائقية (Relational Database) من حيث مكوناتها وأهم خصائصها.

الجدول التالي يمثل معلومات الطالب (Student) في قاعدة بيانات إحدى الجامعات

- اسم الجدول Student
- كل صف يمثل معلومات تخص طالباً واحد فقط.
- المفتاح الرئيس للجدول هو St _No كل طالب يجب أن يكون له رقم مختلف عن بقية الطلاب.
- الصفة Dept_Code تمثل القسم الذي ينتمي إليه أي طالب .
- نقطة تقاطع الصفة (Gpa) العمود مع الصف الثالث تمثل المعدل التراكمي للطالب رقم ٢٠٠١ - ٠١ -١٠ .
- مجال القيم:كل صفة يجب أن يكون لها مجال ثابت من القيم فمثلا Gpa يجب أن تحتوي على رقم حقيقي بين ١.. ٥. القسم Dept_Code يجب أن يكون أحد الأقسام الدراسية الموجودة في الجامعة.

المفتاح الرئيسي
عامود
صف

Student				
St _No	St _Name	Dept_Code	Birth_Date	Gpa
2000-01-101	Ali	Comp	12-08-1980	4.2
2001-02-99	Khalid	Math	10-10-1982	3.5
2001-01-10	Sami	Comp	01-01-1981	3.75

معدل الطالب رقم
200-01-10

- لا توجد هناك أهمية لترتيب الصفوف أو الأعمدة.فمثلا يمكن أن يكون الجدول السابق على الشكل التالي:

Student				
St _No	St _Name	Gpa	Birth_Date	Dept_Code
٢٠٠١-٠١-١٠	Sami	٣,٧٥	٠١-٠١-١٩٨١	Comp
٢٠٠١-٠٢-٩٩	Khalid	٣,٥	١٠-١٠-١٩٨٢	Math
٢٠٠٠-٠١-١٠١	Ali	٤,٢	١٢-٠٨-١٩٨٠	Comp

مفاتيح الجداول (العلاقات):

تعتبر المفاتيح من أهم خصائص قواعد البيانات العلائقية حيث إنها تكون المميز لجدول معين من جهة والرابط الذي يربط الجداول المختلفة مع بعضها من جهة أخرى . ويمكن تقسيم المفاتيح في قواعد البيانات العلائقية إلى عدة أقسام :

أ - **المفتاح الأعظم** (Super Key): وهو أقل مجموعة من الصفات التي يمكن أن تميز الصف في الجدول عن بقية الصفوف الأخرى .فمثلا هذه المجموعة من الصفات يمكن أن تكون مفتاحا أعظم.

St_No
St_No, St _Name
St_No ,dept_code

ب -**المفتاح المرشح** (Candidate Key) : وهو الصفة (مجموعة الصفات) التي يمكن اختيارها كمفتاح رئيس للجدول ويجب أن يكون هناك أكثر من صف له نفس القيمة لهذه الصفة أو الصفات وكذلك يجب أن يكون له قيمة (ليس Null) .

ولكن كما لاحظنا فإن St_No, St _Name هي مفتاح أعظم ولكنه ليس مفتاحا مرشحا ليكون مفتاحا رئيسا لأن St_No وحدة يكفي لتمييز أي صف عن بقية الصفوف، ،لذلك فإن St_No يعتبر مفتاحا مرشحا ليكون مفتاحا رئيسيا .

ج - **المفتاح الرئيس** (Primary Key) : وهو المفتاح الذي تم اختاره من مجموعة المفاتيح المرشحة ليكون محددا لكل صف في الجدول .يمكن أن نختار St_No ليكون مفتاحاً رئيسا .

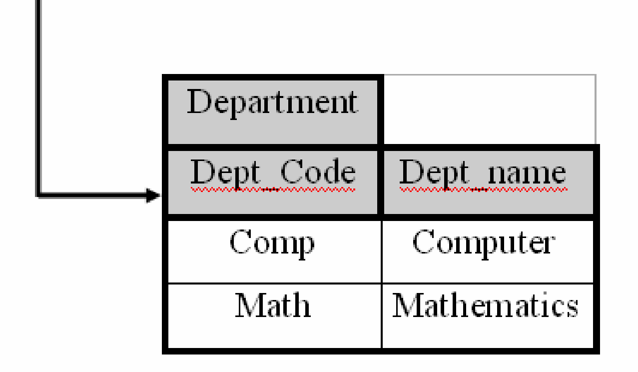
د - **المفتاح الثانوي** : هو عبارة عن صفة أوصفات تستخدم لغايات الاسترجاع ،فمثلا لو كان لدينا جدول يحتوي على قائمة بالعملاء فالمفتاح الرئيس هو رقم العميل Customer _id

ولكن إذا أردنا أن نسترجع رقم هاتف عميل معين (ولكن من سيحفظ أرقام العملاء ؟!) ففي هذه الحالة عادة ما يستخدم الاسم في عملية البحث وليس الرقم ،فيتم اختيار اسم العميل كمفتاح ثانوي .

Address	tel	Customer name	Customer _id

هـ - **المفتاح الأجنبي** (Foreign Key) : وهو صفة أو صفات تشير إلى مفتاح رئيس أو قيمة غير مكررة(Unique) في جدول آخر فمثلا تمل الصفة (Dept_Code) في جدول المتدرب (Student) مفتاح أجنبيا(Foreign Key) لجدول الأقسام (Department)

Student				
St _No	St _Name	Gpa	Birth_Date	Dept_Code
2001-01-10	Sami	3.75	01-01-1981	Comp
2001-02-99	Khalid	3.5	10-10-1982	Math
2000-01-101	Ali	4.2	12-08-1980	Comp

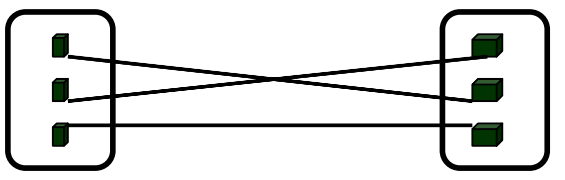


Department	
Dept_Code	Dept_name
Comp	Computer
Math	Mathematics

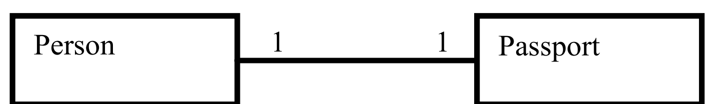
التشاركية بين الجداول (العلاقات):

وتمثل الدرجة التي ترتبط بها الجداول مع بعضها فيجب أن تحدد هذه الروابط بشكل واضح لمعرفة كيفية ارتباط هذه الجداول مع بعضها . هناك ثلاث درجات لارتباط الجداول :

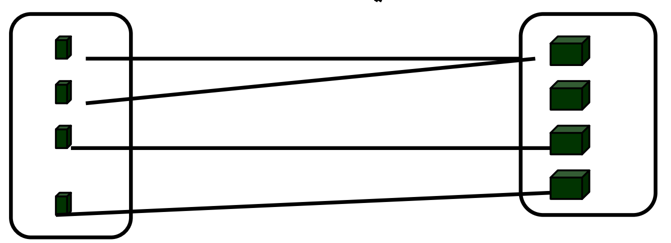
١. **واحد ــ واحد (١:١):** وهذا يعني أن قيمة واحدة في الجدول الأول تقابل قيمة واحدة فقط في الجدول الثاني



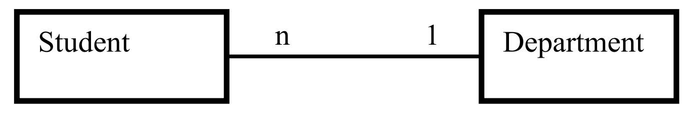
فمثلا يمكن أن نحدد على سبيل المثال أن لكل شخص جواز سفر واحد فقط وأن جواز السفر يعود لشخص واحد فقط .



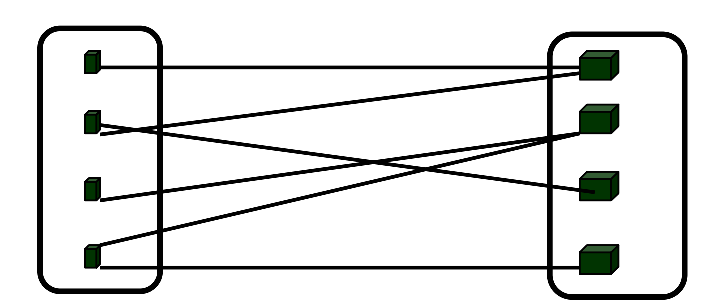
٢. **واحد ــ متعدد أو متعدد ــ واحد** (١: N أو N :١) وهـذا يعني أن قيمة في الجدول الأول تقابل قيمة في الجدول الثاني وأن القيمة في الجدول الثاني يمكن أن يقابلها قيمة أو أكثر في الجدول الأول.



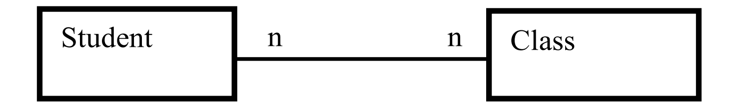
فمثلا يجب أن يتبع المتدرب لقسم واحد فقط وفي الوقت نفسه يمكن أن يكون هنالك أكثر من طالب ينتمي لهذا القسم .



٣. **متعدد ــ متعدد(N:N)** : وهذا يعني أن قيمة في الجدول الأول تقابل قيمة أو أكثر في الجدول الثانِي وأن القيمة في الجدول الثاني يمكن أن يقابلها قيمة أو أكثر في الجدول الأول.



فمثلا يمكن للطالب أن يسجل في أكثر من شعبة وكذلك الشعبة يمكن أن يسجل فيها أكثر من طالب.



تمارين

١ - عرف ما يلي :

- المفتاح المرشح
- المفتاح الرئيس
- المفتاح الأجنبي

٢ - وضح باستخدام الرسم علاقة الإشراف بين المدرس و الطالب.

٣ - وضح باستخدام الرسم نوع العلاقة بين القسم ومجموعة الاختصاصات في ذلك القسم في مستشفى.

٤ - أعط مثالا مع الرسم نوع التشاركية في قاعدة بيانات مستشفى

- واحد ـ واحد (N: N)
- واحد ـ متعدد (١:N)
- متعدد ـ متعدد (N: N)



المملكة العربية السعودية
المؤسسة العامة للتعليم الفني والتدريب المهني
الإدارة العامة لتصميم وتطوير المناهج

تصميم قواعد البيانات

نموذج الكيانات والعلاقات

نموذج الكيانات والعلاقات



الجدارة:

معرفة عناصر نموذج العلاقات والكيانات والقدرة على تحويل ناتج عملية التحليل لنظام إلى نموذج العلاقات والكيانات.

الأهداف:

- أن يتعرف التدرب على عناصر نموذج العلاقات والكيانات .
- أن يحول المتدرب ناتج عملية التحليل للنظام إلى نموذج مفاهيم .

مستوى الأداء المطلوب:

تحويل ناتج عملية التحليل للنظام إلى نموذج مفاهيم نسبة ١٠٠٪.

الوقت المتوقع للتدريب:

٤ ساعات .

الوسائل المساعدة:

قلم + دفتر

متطلبات الجدارة:

أن يكون المتدرب قد أتقن الجدارة في الوحدات السابقة .

مقدمة :

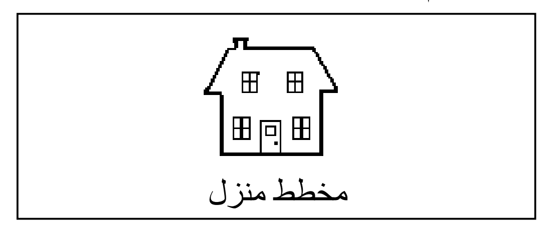
إن هدف عملية التصميم هو الوصول إلى فهم صحيح للنظام للمساعدة في عملية تطوير هذا النظام، وهذا ليس بالأمر السهل إذ لابد من وجود مقياس صحيح للحكم على هذا الفهم. ومن هنا برزت الأهمية لاستخدام العديد من الأدوات التي تساعد المصمم لوضع التصور والفهم الصحيحين لعمل هذا النظام.ومن هذه الأدوات استخدام النماذج التمثيلية التي تصف مكونات النظام وكيفية ارتباطها مع بعضها.وسنقوم في هذا الفصل بدراسة كيفية تمثيل البيانات باستخدام **نموذج الكيانات والعلاقات** Entity **Relationship (ER)Diagram**.

النماذج:

ما هو النموذج ؟

النموذج عبارة عن وصف رسومي (تمثيلي) لوصف الحقائق التي لا يمكن رؤيتها مباشرة.

وبعبارة أخرى هو وصف مجرد للكائنات الحقيقة. **نموذج البيانات** هو عبارة عن تمثيل بسيط لوصف تراكيب البيانات المعقدة في واقع الحياة العملية على شكل رسومي دون النظر إلى مكان وكيفية تخزين أو الوصول إلى هذه البيانات.ويستخدم هذا النموذج كوسيلة اتصال مابين المصمم من جهة وبين المبرمجين والمستخدمين من جهة أخرى. إذ حتى لو كان لدينا العديد من المبرمجين المحترفين فلا نستطيع الحصول على نظام جيد دون أن يكون هذا النظام قد صمم بشكل صحيح .والشكل التالي يبين مواصفات لمنزل وهذا الشكل يكون كوسيلة اتصال مابين الشخص الذي يرغب في بناء المنزل (الزبون) وكذلك بين المهندس(المصمم) من جهة وبين المقاول (المنفّذ) الذي سيقوم ببناء المنزل،وفي بناء أنظمة قواعد البيانات يمثل الزبون صاحب النظام ويمثل المصمم (مصمم قاعدة البيانات) والمقاول المنفذ هو مجموعة المبرمجين التي تقوم ببناء النظام .


مخطط منزل

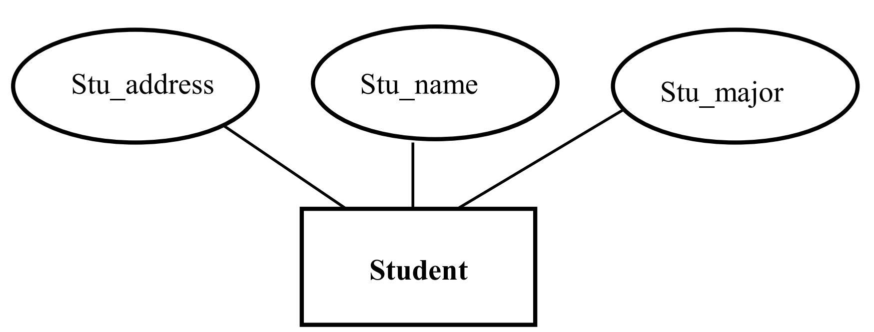
نموذج الكيانات والعلاقات:

هو عبارة عن نموذج لتمثيل كيانات النظام وصفاتها وكيفية ارتباط هذه الكيانات مع بعضها باستخدام رموز رسومية..ولنتعرف الآن على عناصر هذا النموذج:

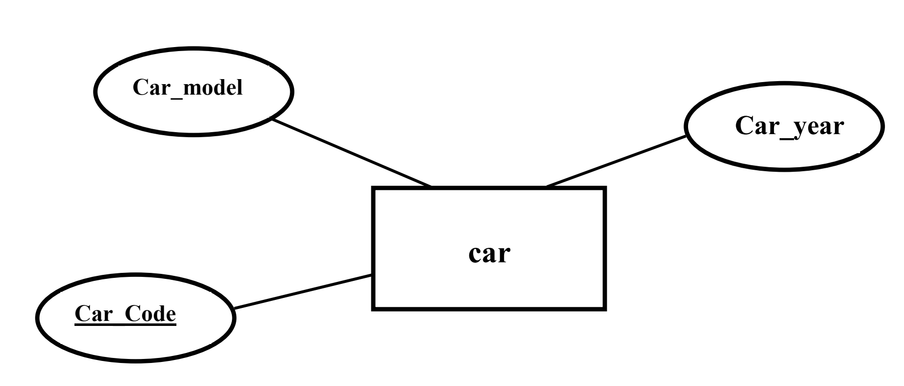
مجموعة الكيانات (Entity Set) وتمثل المجموعة التي تنتمي إليها مجموعة الكائنات(Objects) المتشابهة وتمثل بجدول في قاعدة البيانات العلائقية . و **الكيان**(Entity) هو عبارة عن كائن أو شيء محط الاهتمام في النظام وعلينا أن نقوم بجمع وتسجيل البيانات عن هذا الكيان. مثلا المتدرب ، المقرر، المدرس و الشعبة تعتبر كيانات مهمة في نظام قاعدة البيانات لجامعة .ويمثل الطبيب و المريض و وصفة العلاج كيانات مهمة في قاعدة بيانات لمستشفى . ويرمز لمجموعة الكيانات بمستطيل يحتوي على اسم الكيان .



الخصائص أو الصفات (Attributes): هي عبارة عن الصفات المميزة للكيان، وبعبارة أخرى هي المعلومات الواجب تخزينها عن كائن معين وتمثل بأعمدة الجدول في قاعدة البيانات العلائقية..فمثلا لكل طالب يجب أن نسجل الاسم، الرقم، تاريخ الميلاد ، التخصص، ولمنتج معين يكون الرقم الوصف، الطول، العرض، اللون.ويرمز للصفة بشكل بيضاوي يحتوي على اسم الصفة وتربط الصفة مع الكيان بواسطة خط مستقيم.



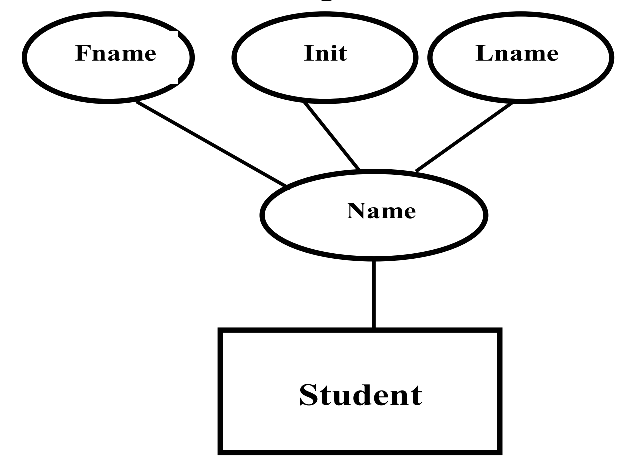
ولكل صفة يجب أن نحدد **مجال القيم (Domain)**: وهو مجموعة القيم لهذه الصفة فمثلا رقم المتدرب يجب أن يكون عدداً صحياً من عشر خانات، واسم المتدرب يجب أن يحتوي على قيم رمزية بطول ٣٠ حرف، والمعدل التراكمي يجب أن يحتوي على عدد كسري مابين ٠.. ٥ مثلا (٣,٥). تاريخ الميلاد يجب أن يكون مقبولاً بحيث لا يتجاوز عمر المتدرب عند القبول ٢٢ سنة. وبعض الصفات يمكن أن تشترك في نفس مجال القيم فمثلا القسم الدراسي للطالب والمدرس يكون اسماً من أسماء الأقسام في الجامعة. والصفة (مجموعة الصفات) التي تم اختيارها كمفتاح رئيس **(primary key)** تُمثل كأي صفة ولكن يوضع خط تحت الاسم.



وفي عملية تحديد الصفات للكيانات لابد من أن نحدد

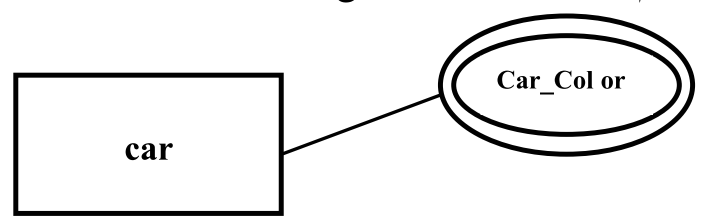
أ - الصفات البسطة والمركبة **Simple and Composite Attributes** :

وتقسم إلى صفات بسيطة أي لا يمكن تجزئتها مثل رقم الطالب، الجنس تاريخ الميلاد. أو مركبة أي يمكن تجزئتها كالاسم (الاسم الأول، الثاني، واسم العائلة)، العنوان (المدينة، الحي، الشارع، رقم المنزل). .ويرمز للصفة المركبة بشكل بيضاوي ترتبط معه أشكال بيضاوية أخرى يحتوي كل منها على اسم الصفة الفرعية وتربط الصفات الفرعية مع الصفة الرئيسة بواسطة خط مستقيم .



ب -صفات وحيدة أو متعددة القيم **Single-Valued or Multiple-Valued Attributes** :

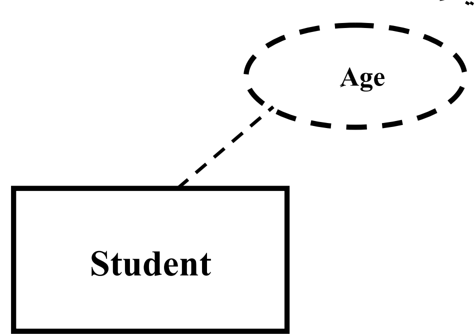
الصفات التي تحتوي على قيمة واحدة مثل (رقم السيارة، تاريخ الصنع) أو عدة قيم مثل لون السيارة (فيمكن أن يكون هناك لون للسقف، الجسم، الجوانب) وكذلك يمكن أن يكون للمدرس أكثر من رقم هاتف أو أكثر من بريد إلكتروني. ويرمز للصفة متعددة القيم بشكل بيضاوي داخل شكل بيضاوي آخر يحتوي على اسم الصفة وتربط الصفة مع الكيان بواسطة خط مستقيم.



ج ـ الصفات المشتقة (**Derived Attributes**):

وهي الصفات التي يمكن اشتقاقها من صفات أخرى ويرمز لها بشكل بيضاوي متقطع يحتوي على اسم الصفة وترتبط مع الكيان بخط مستقيم متقطع أيضا كما في الشكل التالي. مثل عُمر المتدرب يمكن حسابه على أنه الفرق بين تاريخ الميلاد والتاريخ الحالي.

العمر = التاريخ الحالي – تاريخ الميلاد.

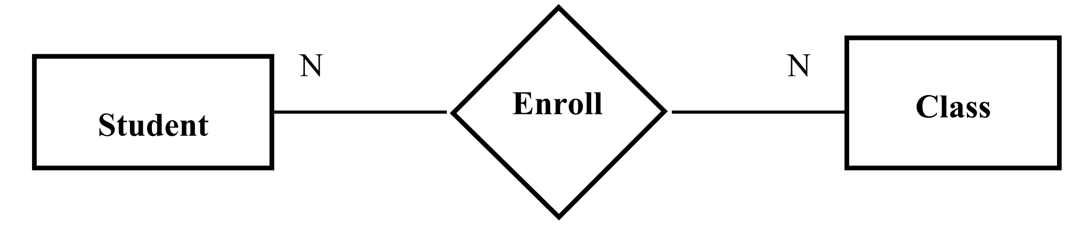


الصفات المشتقة يجب أن لا تخزن ولكن توضع طريقة لحسابها عند عملية الاسترجاع.ولكن قد نخزن بعض الصفات المشتقة إذا كانت عملية حسابها تأخذ وقتا كبيرا وفي نفس الوقت يتم طلبها بشكل كبير مثل المعدل التراكمي للطالب.

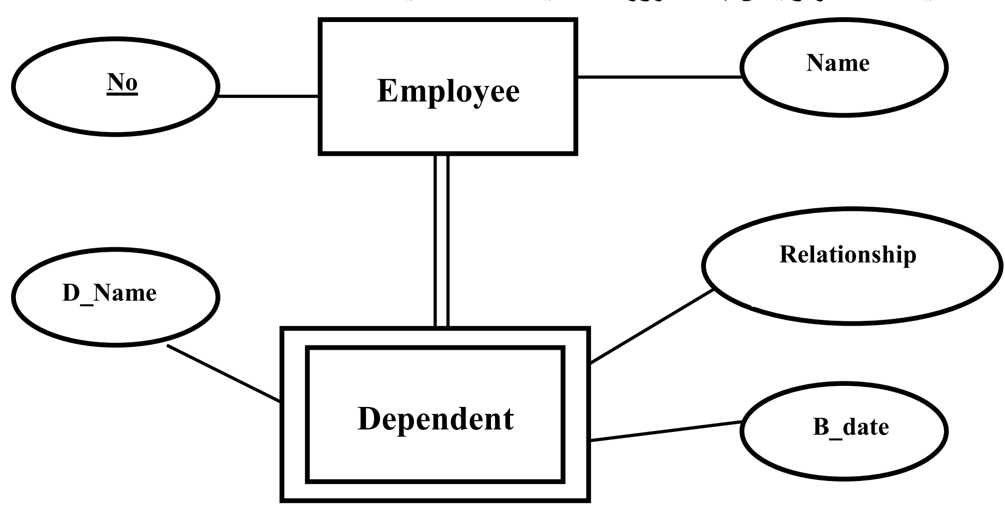
الروابط أو العلاقات (Relationships):

وهي عبارة عن الرابط أو العلاقة مابين الكيانات واسم هذه الرابطة يجب أن يعبر عن كيفية هذا الترابط ويكون على شكل فعل (ينتمي، يحتوي، يسجل، يتكون من....). ويرمز لها بشكل معين يحتوي على اسم الرابط أو العلاقة.وكذلك لكل علاقة درجة تشاركية. وتبين مقدار التشارك مابين الكيانات إما **واحد ــ واحد (١:١) أو واحد ــ متعدد(N :1) أو متعدد ــ متعدد (N:N).**

فالطالب يسجل في شعبة أو أكثر والشعبة يسجل فيها مجموعة من الطلاب .



الكيانات الضعيفة: وهي عبارة عن الكيانات التي لا توجد مستقلة بنفسها في النظام وبعبارة أخرى فإن وجودها يعتمد على وجود كيان آخر فمثلا لنفرض أن مؤسسة ما تسجل معلومات عن أسماء الأشخاص التابعين للموظف مثل الأبناء، الزوجة أو الوالدين. فوجود معلومات التابع مرتبط بوجود الموظف وفي هذه الحالة يختار المفتاح الرئيس للكيان الرئيس مع صفة من صفات التابع (مثل الاسم) لتشكل مفتاحا رئيسا للكيان التابع ويوضع تحته خط مقطع. ويرمز للكيان الضعيف بمستطيل داخل مستطيل يحتوي على اسم الكيان الضعيف ويرتبط مع الكيان الرئيس بخطين مستقيمين (يعني أن وجود الكيان الأول شرط لوجود الكيان الآخر وليس بالضرورة للكيانات الضعيفة فقط).

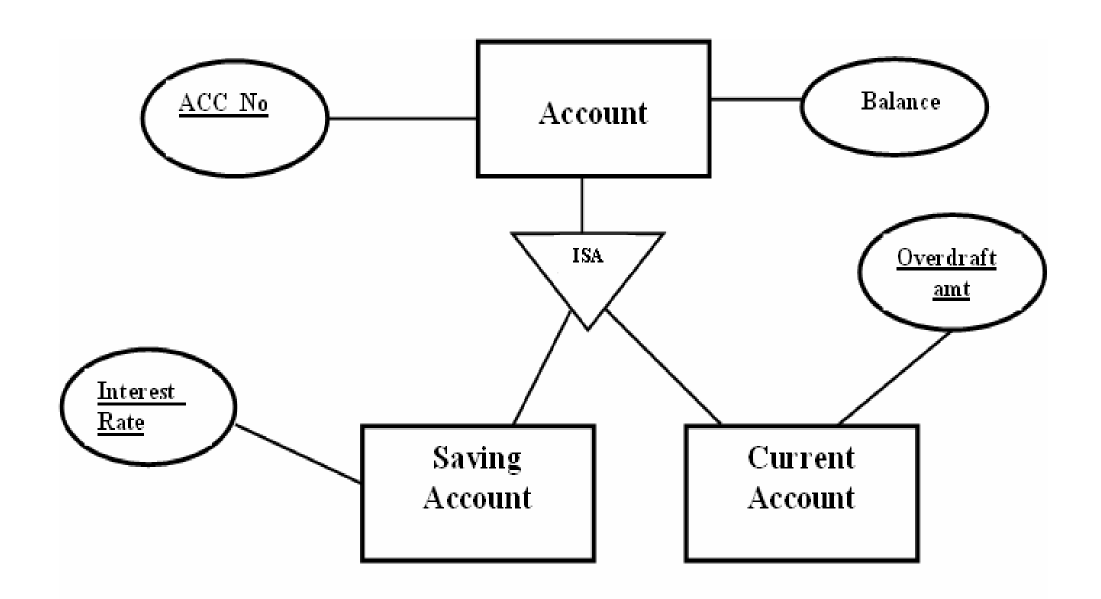


تمثيل الأنواع الرئيسة والأنواع الفرعية (Supertype and Subtype) :

هناك بعض الكيانات الفرعية التي تتبع إلى نوع رئيس (أعلى) Supertype فمثلا بالنسبة للحساب البنكي يمكن أن يكون هناك أكثر من نوع للحسابات ولكن جميع هذه الحسابات تشترك في الكثير من الصفات ففي هذه الحالة نقوم بإنشاء كيان الحساب البنكي Account بحيث يحتوي على جميع هذه الصفات ،ثم بعد ذلك نقوم بإنشاء كيانات فرعية للحسابات يحتوي كل منها على الصفات الخاصة بهذا النوع فقط.

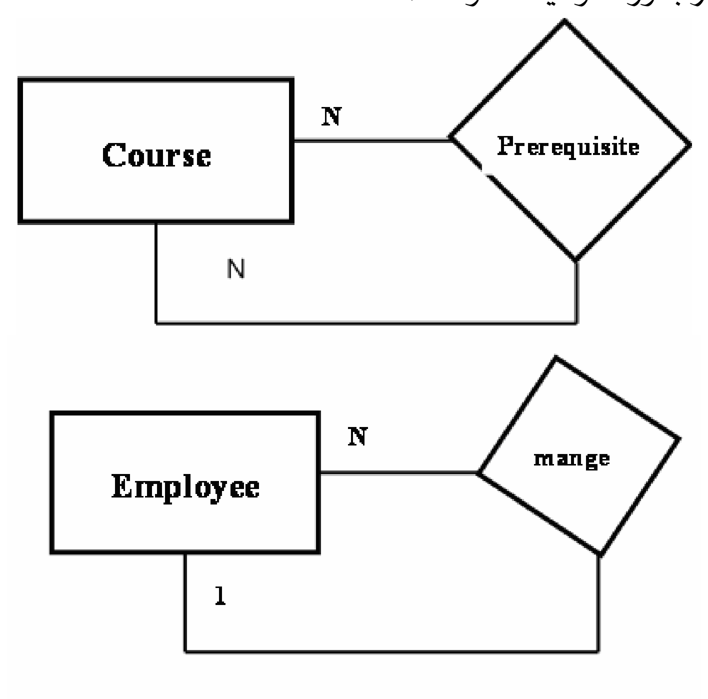
مثال: لنفرض أن لكل الحساب حقل يمثل رقم الحساب وحقل يمثل الرصيد الحالي وفي نفس الوقت لدينا نوعين من الحسابات: الحساب الجاري (Current Account) وفيه الصفة (Overdraft Amount) وهي أعلى قيمة يسمح لصاحب الحساب أن يسحبها عندما لا يكون لديه رصيد. والنوع الثاني حساب التوفير وفية صفة معدل الفائدة(Interest Rate) .

وتمثل العلاقة بين الأنواع الرئيسة العليا والأنواع الفرعية بمثلث مقلوب يحتوي على (ISA) بمعنى يكون.



تمثيل علاقة الكيان مع نفسه (Recursive):

وفي هذه الحالة نبين كيفية تمثيل ارتباط الكيان مع نفسه، فمثلا نفرض أن المقرر الدراسي يمكن أن يكون لديه متطلب سابق أو أكثر (وهذا المتطلب هو عبارة عن مقرر) وكذلك يجب أن يكون للموظف مدير واحد فقط (والمدير بدوره هو أيضا موظف)



حالـة دراسية: سنقوم في هذا المثال بعملية تحويل عملية تحليل شركة ما إلى نموذج مفاهيم (نموذج الكيانات و العلاقات ER Diagram). حيث إن الشركة تهتم بتسجيل معلومات عن الأقسام والمشاريع التي تنفذها الشركة وكذلك عن الموظفين العاملين فيها والتابعين لهؤلاء الموظفين .

١ - تقسم الشركة إلى عدة أقسام ولكل قسم اسم واحد ورقم (لا يجوز أن يكون هناك أكثر من قسم بنفس الاسم أو الرقم)، لكل قسم موظف يدير هذا القسم ويجب معرفة التاريخ الذي بدأ فيه هذا الموظف بإدارة القسم ،ولكل قسم موقع واحد فقط.

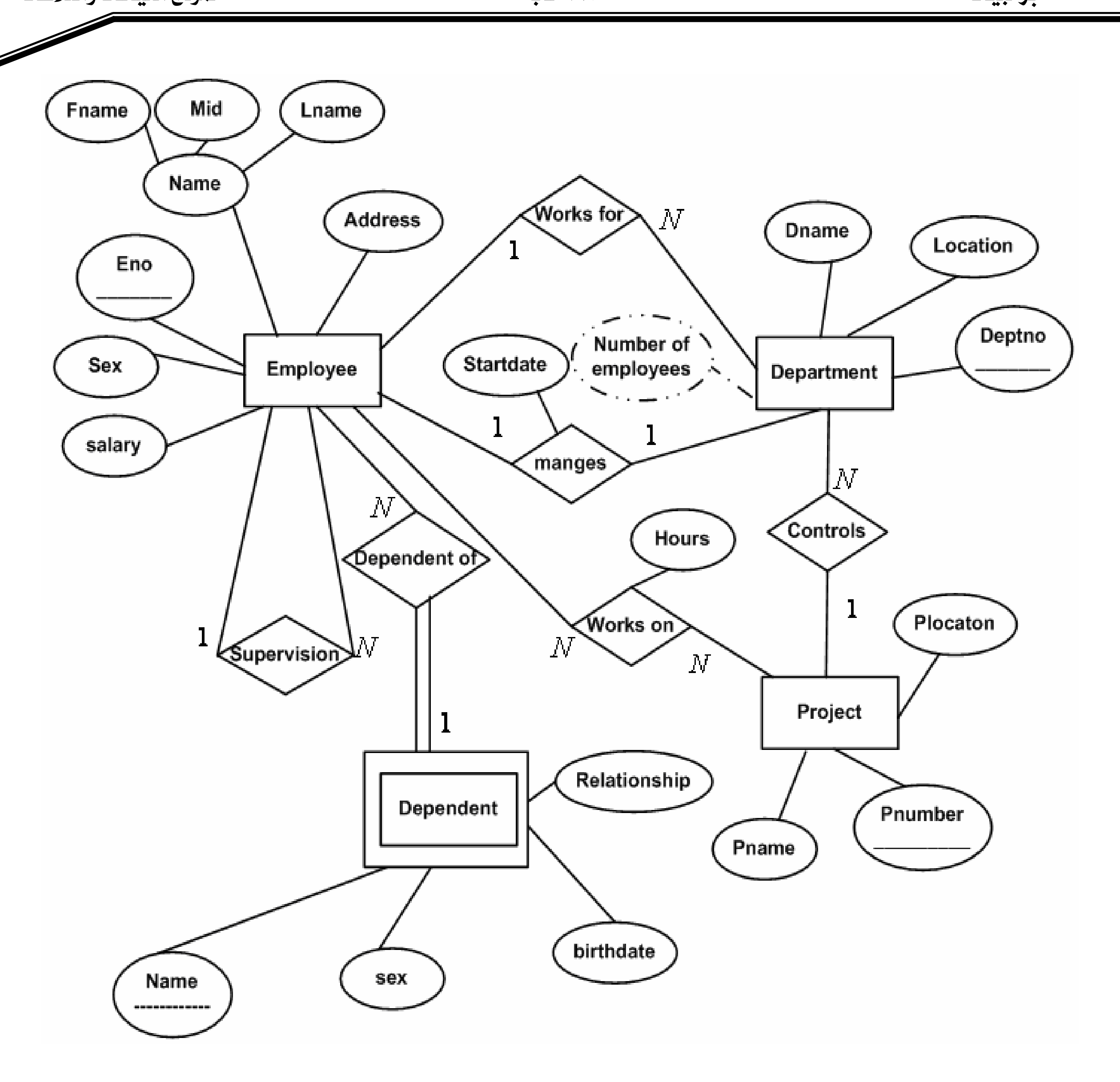
٢ - القسم يمكن أن يدير عدة مشاريع ولكل مشروع رقم واسم ومكان تنفيذ.

٣ - يمكن أن يعمل في القسم موظف أو أكثر ولكن الموظف يجب أن يتبع لقسم واحد فقط ونحتفظ بالمعلومات التالية عن الموظف (الرقم لكل موظف رقم يميزه عن بقية الموظفين، الاسم (الاسم الأول، الاسم الثاني واسم العائلة)، الجنس ، الراتب والعنوان.

٤ - الموظف يمكن أن يعمل في عدة مشاريع (وليس بالضرورة أن يدار المشروع من نفس القسم الذي يتبع إليه الموظف) ونحتفظ بعدد الساعات التي عملها الموظف في كل مشروع.

٥ - تحتفظ الشركة بمعلومات عن التابعين لكل موظف مثل الاسم ،تاريخ ، الميلاد ، الجنس، صلة القرابة .

٦ - تهتم الشركة بمعرفة عدد الموظفين الذين يتبعون لقسم معين.


Fname
Mid
Lname
Name
Address
Eno
Sex
salary
Employee
Works for
1
N
Dname
Location
Deptno
Department
Startdate
Number of employees
manges
1
1
N
Controls
1
Hours
Works on
N
N
Dependent of
N
1
Supervision
1
N
Project
Plocaton
Pname
Pnumber
Dependent
Relationship
Name
sex
birthdate

تمارين

١ -عرف ما يلي:
- الكيان Entity .
- الصفة Attribute .
- العلاقة Relationship .

٢ -اذكر أنواع التشاركية بين الكيانات مع ذكر الأمثلة واستخدام الرسم .

٣ - ما الفرق بين الصفات وحيدة القيم والصفات متعددة القيم؟ وكيف تُمثل باستخدام الرسم ؟

٤ -اذكر مثالاً على الكيانات الضعيفة وارسم نموذج مفاهيم لتوضيح ذلك .

٥ - ارسم نموذج العلاقات والكيانات لكل مما يلي :

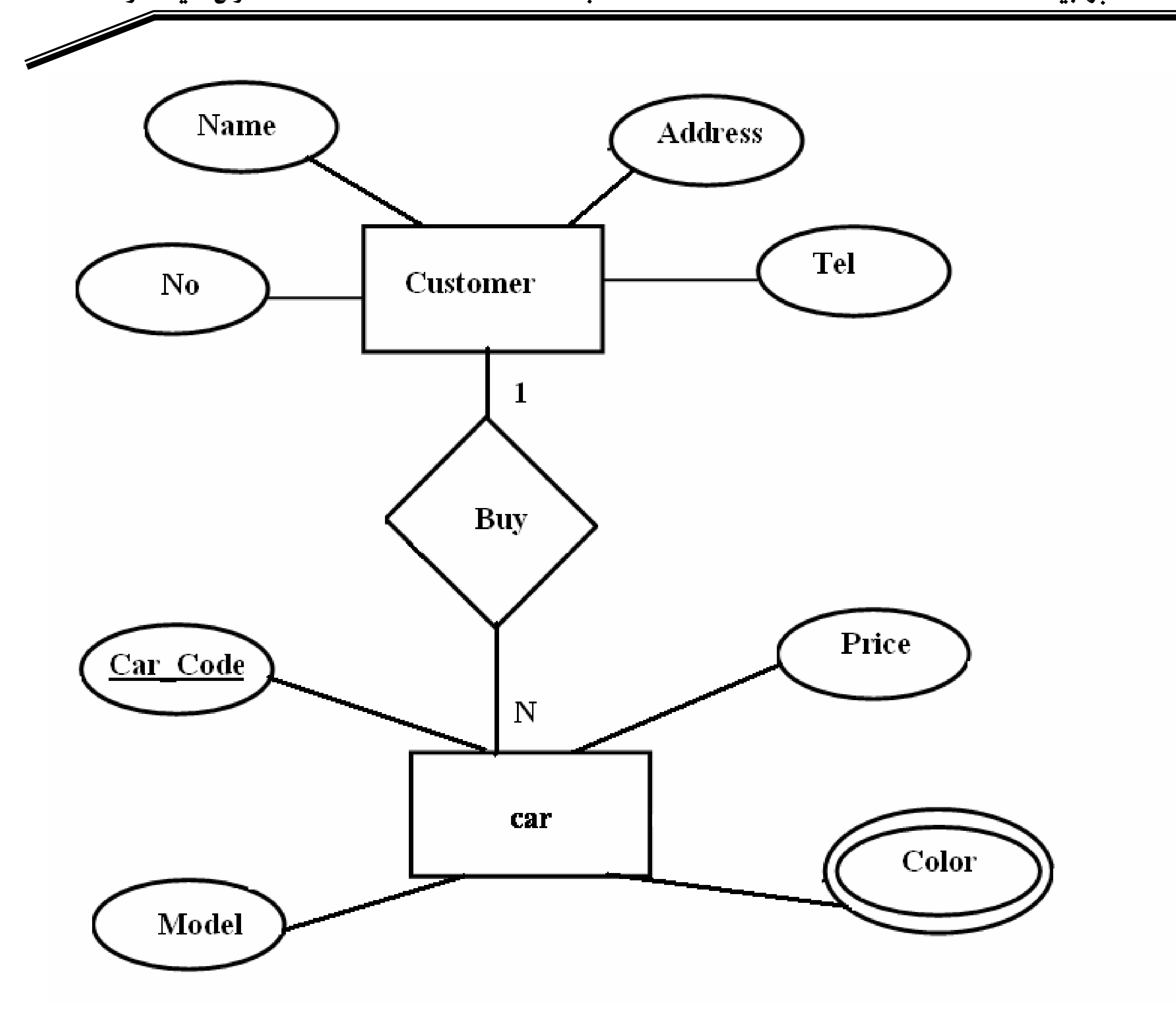
أ - في قاعدة بيانات لمكتبة المؤلف يمكن أن يؤلف أكثر من كتاب والكتاب يمكن أن يشترك في تأليفه أكثر من مؤلف ،وكذلك يجب أن يتبع الكتاب لموضوع واحد فقط.

ب -في قاعدة بيانات لمستشفى يمكن أن يشرف الطبيب على أكثر من مريض والمريض يجب أن يشرف عليه طبيب واحد .

ج - في قاعدة بيانات لجامعة يمكن أن يدرس المدرس أكثر من شعبة والشعبة تكون لمقرر واحد فقط ويجب أن يدرسها مدرس واحد .

٦ - ارسم نموذج العلاقات والكيانات كاملاً لنظام مبيعات بحيث يمكن للزبون أن يشتري أكثر من منتج ويجب أن تتم عملية الشراء من خلال فاتورة والفاتورة تحرر من قبل موظف واحد فقط وكذلك فإن لكل منتج كمية معينة داخل المحل وكمية أخرى في المستودع وفي حال نفاذ الكمية من المحل يقوم بطلب كمية أخرى من المستودع والمستودع بدورة يقوم بتزويد المحل بمنتج أو أكثر في نفس الوقت.

٧ - صف على شكل نقاط نموذج العلاقات والكيانات التالي :


Name
Address
No
Customer
Tel
1
Buy
N
Car_Code
Price
car
Color
Model

تصميم قواعد البيانات

الصيغ المعيارية

الصيغ المعيارية

٤

الجدارة:

القدرة على تحويل الجداول إلى الصيغة المعيارية الثالثة **3NF** .

الأهداف:

- أن يتعرف المتدرب على مشاكل تكرار البيانات(Data Anomalies) :
- أن يتعرف المتدرب على الاعتمادية الوظيفية
- أن يستطيع المتدرب تعريف الصيغة المعيارية الأولى
- أن يستطيع المتدرب تعريف الصيغة المعيارية الثانية
- أن يستطيع المتدرب تعريف الصيغة المعيارية الثالثة

مستوى الأداء المطلوب:

أن يتقن المتدرب عملية تحويل الجداول إلى الصيغة المعيارية الثالثة **3NF** بنسبة١٠٠٪.

الوقت المتوقع للتدريب:

ساعتان

الوسائل المساعدة:

قلم + دفتر

متطلبات الجدارة:

أن يكون المتدرب قد أتقن الجدارة في الوحدات السابقة .

مقدمة :

إن عملية وضع تصميم قاعدة البيانات في الصيغة المعيارية يشكل لبنة أساسية في عملية التصميم الجيد لقاعدة البيانات.وتتم هذه العملية على عدة مراحل يتم خلالها تخليص قاعدة البيانات من التكرار غير المسوغ للبيانات بالاعتماد على قوانين الاستنتاج والاعتمادية الوظيفية. و.سنقوم في هذا الفصل بالتعرف على الشروط و القوانين اللازمة للوصول بقاعدة البيانات إلى المستوى المعياري الثالث(Third Normal Form **3NF**).

مشاكل تكرار البيانات (Data Anomalies) :

Employee_department						
Empno	Ename	Job	Salary	Deptno	Dname	Loc
101	Sami	clerk	3000	10	Accounting	Riyadh
205	Khalid	manager	2500	10	Accounting	Riyadh
303	Ali	salesman	1200	20	Sales	Jeddah
502	Saeed	salesman	2100	20	Sales	Jeddah
601	Salem	clerk	1000	30	Operation	Dmmam

نلاحظ في الجدول السابق أن معلومات الموظف والقسم الذي يعمل فيه موجودة في جدول واحد ونتيجة ذلك تكرار بعض البيانات مثل اسم وموقع القسم في كل سجل وهذا يؤدي إلى عدة مشاكل :

١ - **مشكلة الإضافة** :أي إننا لا نستطيع أن نظيف قسماً جديداً إلا إذا كان القسم يحتوي على، موظف ،لأن المفتاح الرئيس للجدول هو رقم الموظف.

٢ - **مشكلة التعديل** : نلاحظ تكرار اسم وموقع القسم فإذا قمنا بتعديل موقع (Loc)القسم رقم ٢٠ من Jeddah إلى Riyadh فلابد من إجراء عملية التعديل لجميع الموظفين في هذا القسم وإلا ستؤدي هذه العملية إلى عدم توافقية البيانات أي نفس رقم القسم ولكن أكثر من موقع . وكذلك إذا تمت عملية التغيير عند الموظف رقم ٣٠٣ عن طريق الخطأ . وبالتالي لو قمنا بعملية استرجاع لجميع الموظفين الذين يعملون في Jeddah فإن الموظف رقم ٣٠٣ لن يظهر بين الموظفين .

٣ - **مشكلة الحذف** : نلاحظ أن القسم رقم ٣٠ يحتوي على موظف واحد فقط ،فلو قمنا بحذف الموظف رقم ٦٠٦ فإن معلومات القسم رقم ٣٠ سوف تختفي من الجدول .

الاعتمادية الوظيفية (Functional Dependency FD):

وهي اعتماد قيمة إحدى صفات الكيان على قيمة صفة(صفات) أخرى ويرمز لها بالرمز (⟵)

مثال $A \longrightarrow B$

يعني أن B تعتمد اعتمادا وظيفيا على A وهنا نستطيع أن نقول أن قيمة A تحدد قيمة B . ومن خلال تحديد الاعتمادية نستطيع أن نحدد المكان الذي يجب أن توضع فيه الصفة وهذا بالتالي يؤدي إلى وضع البيانات في المكان الصحيح ونتخلص من عملية تكرار البيانات وما يترتب على التكرار من مشاكل(Anomalies) .

مثال: لكل موظف اسم واحد فقط ولكل موظف قسم واحد يعمل فيه إذا :

FD1 : Empno ⟶ Ename
FD2 : Empno ⟶ Deptno

ويمكن أن نعيد كتابة هذه الاعتمادية على الشكل التالي

FD1 : Empno ⟶ Ename,Deptno

FD :Functional Dependency

قواعد الاستنتاج

وهي عبارة عن مجموعة من القواعد تستخدم في عملية تحديد الاعتمادية الوظيفية (Functional Dependency FD) وتتلخص هذه القواعد بستة قواعد على النحو التالي :

١ - الانعكاسية **Reflexive** :إذا كانت Y جزء من X ((Y محتواه في X)

فإن X تحدد Y

$1\text{-}\ X \supseteq Y:\quad X \rightarrow Y$

٢ - قاعدة الزيادة أو الإضافة **Augmentation** :إذا كان X تحدد Y فإن إضافة Z إلى X تعني أنه بالإمكان إضافة Z إلى Y

$2\text{-}\ \{X \rightarrow Y\} \models XZ \rightarrow YZ$

٣ - قاعدة التعدي **Transitive** : تعني أنه إذا كانت X تحدد Y وكانت Y تحدد Z فإن X تحدد Z .

$3\text{-}\ \{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$

٤ - قاعدة الاتحاد **Union** : تعني أنه إذا كانت X تحدد Y و X تحدد Z فإننا نستطيع أن نقول أن X تحدد YZ .

4- {X→ Y ,X→Z} ⊨ X →YZ

٥ - قاعدة التقسيم Decomposition وهي عكس قاعدة الاتحاد

5- {X→ YZ } ⊨ X →Y , X → Z

٦ - قاعدة التعدي الزائف **Pseudotransitive** تشبه قاعدة التعدي

6- {X→ Y , WY→Z } ⊨ WX →Y

⊨ تعني أنه إذا تحقق الطرف الأيسر فإننا نستطيع استنتاج الطرف الأيمن .

تعريف الصيغة المعيارية الأولى(First Normal Form 1NF):

نستطيع أن نقول أن الجدول في الصيغة المعيارية الأولى إذا كانت جميع أعمدة الجدول تحتوي على بيانات بسيطة أو مفردة(غير مركبة) أي إن كل عمود يحتوي على قيمة واحدة فقط .

مثال ١ يمثل الجدول التالي معلومات موظف Employee:

No	Name			Adresse		
	Fname	Mid	Lname	city	Street	House no
100	Ali	Salem	musa	Riyadh	Immam saud	210
120	Saeed	Eisa	Ali	Riyadh	King Fahad	202

نلاحظ في الجدول أن الاسم يتكون من ثلاثة أجزاء وكذلك العنوان فبالتالي لا نستطيع أن نخزن قيمة واحدة في عمود الاسم وكذلك بالنسبة للعنوان وهذا يخالف شروط قاعدة البيانات بأن العمود يجب أن يحتوي على قيمة واحدة فقط.وهذا يعني أن الجدول السابق لا ينطبق عليه شرط الصيغة المعيارية الأولى **1NF,** ولوضع الجدول في الصيغة المعيارية الأول يجب تقسيم الأعمدة المركبة إلى أعمدة بسيطة

No	Fname	Mid	Lname	city	Street	House no
100	Ali	Salem	musa	Riyadh	Immam saud	210
120	Saeed	Eisa	Ali	Riyadh	King Fahad	202

لقد قمنا بتقسيم الأعمدة المركبة إلى أعمدة بسيطة وبالتالي نستطيع أن نقول أن الجدول الآن في الصيغة المعيارية الأولى **1NF** .

مثال ٢ : يمثل الجدول التالي سجل ساعات العمل HOURS لموظف في عدد من المشاريع PROJECTS والقسم الذي يشرف على تنفيذ المشروع

NO	Name	Project_Code	Hours	Deptno	Dname
210	Ali	P1,p2,p3	12,20,40	10,20,20	Research, Operation, Operation
201	Salem	P1,p3	30,15	10,20	Research Operation
305	Ali	P2,p3	40,20	20,20	Operation, Operation

كما هو مبين في الجدول السابق فإن هناك عدداً من الأعمدة تحتوي على أكثر من قيمة مثل رمز المشروع Project_Code وكذلك عدد ساعات العمل Hours والأقسام Deptno التي تشرف على المشاريع . وهذا يعني أن الجدول ليس في الصيغة المعيارية الأولى، ولتحويله يجب أن نقوم بتقسيم الجدول على النحو التالي للتخلص من هذه المشكلة.

NO	Name	Project_Code	Hours	Deptno	Dname
210	Ali	P1	12	10	Research
210	Ali	p2	20	20	Operation
210	Ali	p3	40	20	Operation
201	salem	P1	30	10	Research
201	salem	p3	15	20	Operation
305	Ali	P2	40	20	Operation
305	Ali	p3	20	20	Operation

ولكن تبرز هنا لدينا مشكلة جديدة وهي إيجاد مفتاح رئيسي للجدول إذ أصبح رقم الموظف لا يصلح لأن يكون مفتاحا رئيسا للجدول (Primary Key) لأن من شروط المفتاح الرئيس أن لا يتكرر في أكثر من صف .لنقوم الآن باستخدام الاعتمادية الوظيفية لمحاولة إيجاد المفتاح الرئيس للجدول

FD 1 :No → Name

حيث إن لكل موظف اسم واحد .

FD 2 : Project_Code→ Deptno

حيث إن لكل مشروع قسم واحد يشرف عليه .

FD 3 : Deptno →Dname

حيث إن لكل قسم اسم واحد.

أما بالنسبة لبقية العناصر فمثلا اسم الموظف لا يحدد شيئاً لأنه يوجد هناك أكثر من موظف اسمه Ali فالاسم لا يحدد الرقم وكذلك فإن علي يعمل في أكثر من مشروع .

وكذلك رمز لمشروع لا يحدد عدد الساعات ولاالموظفين الذين يعملون فيه فالمشرع P1 يعمل فيه أكثر من موظف وبساعات مختلفة .

أما بالنسبة للقسم فلا يحدد الموظفين ولا المشاريع فمثلا القسم ٢٠ يشرف على أكثر من مشروع هذه المشاريع يعمل فيها أكثر من موظف .

ففي هذه الحالة يجب علينا القيام بمحاولة جديدة لإيجاد المفتاح الرئيس وتتلخص هذه العملية بمحاولة إيجاد مفتاح مركب(تركيب أكثر من صفة لتشكل المفتاح الرئيس) يقوم بتحديد جميع الصفات في الجدول :

سنقوم بأخذ رقم الموظف مع رقم المشروع

FD 4 :No, Project_Code→ name
FD 5 :No, Project_Code→ Deptno
FD 6 :No, Project_Code→ Hours
FD 7 : Deptno →Dname
FD 8 :No, Project_Code→ Name ,Hours, Deptno, Dname

FD4,FD5 تتطبق من FD1,FD2 حيث إن رقم الموظف وحدة يحدد الاسم وكذلك رمز المشروع يحدد القسم ، أما بالنسبة ل FD5 فإنها تنطبق لأن رقم الموظف ورمز المشروع يحددان عمل الموظف في ذلك المشروع ، وبالتالي نكون قد حصلنا على مفتاح رئيس لهذا الجدول وكذلك قمنا بوضعه في الصيغة المعيارية الأولى (**1NF**).

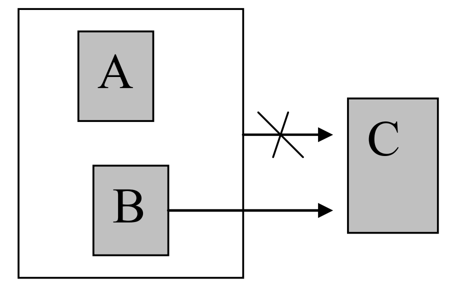
تعريف الصيغة المعيارية الثانية(Second Normal Form 2NF):

نستطيع أن نقول أن الجدول في الصيغة المعيارية الثانية:

١ - إذا كان الجدول في الصيغة المعيارية الأولى.

٢ - إذا لم يحتوي الجدول على اعتمادية جزئية.

الاعتمادية الجزئية: هي أن تعتمد بعض الأعمدة (الصفات) اعتمادا وظيفيا على جزء من المفتاح الرئيس



نلاحظ أن A,B تحدد C أي إن C تعتمد اعتمادا وظيفيا على A,B وكذلك أن B تحدد C أي إن C تعتمد اعتمادا وظيفيا B. وفي هذه الحالة نستطيع أن نقول أن هذا الجدول يحتوي على اعتمادية جزئية .

NO	Name	Project_Code	Hours	Deptno	Dname
210	Ali	P1	12	10	Research
210	Ali	p2	20	20	Operation
210	Ali	p3	40	20	Operation
201	Salem	P1	30	10	Research
201	Salem	p3	15	20	Operation
305	Ali	P2	40	20	Operation
305	Ali	p3	20	20	Operation

والآن هل الجدول السابق في الصيغة المعيارية الثانية ؟

وللإجابة على ذلك نجيب على السؤالين التاليين :

١ - هل الجدول في الصيغة المعيارية الأولى ؟

نعم، لأنه لا توجد هناك قيم متكررة ،كل عمود يحتوي على قيمة واحدة فقط .

٢ - هل توجد هناك اعتمادية جزئية ؟

ولمعرفة ذلك يجب أن نحدد الاعتمادية الوظيفية

FD 1 :No → Name

FD 2 : Project_Code→ Deptno,Dname

FD 3 :No, Project_Code→ name ,deptno, hours

المفتاح الرئيس هو No, Project_Code ولكن No يحدد Name إذا هناك اعتمادية جزئية وكذلك

Project_Code يحدد deptno وDname و هذه اعتمادية جزئية أخرى . وللتخلص من هذه المشكلة يجب أن نقوم بتقسيم الجدول إلى جداول بحيث يضم كل منها الجزء من المفتاح والأعمدة التي تعتمد عليه ونبقي فقط المفتاح المركب مع الأعمدة التي تعتمد عليه:

١ - نقوم بنقل اسم ورقم الموظف إلى جدول جديد ونبقي نسخة من رقم الموظف في الجدول الأصلي (لأنه جزء من المفتاح الرئيس) .

٢ - نقوم بنقل رمز المشروع ورقم القسم إلى جدول جديد ونبقي نسخة رمز المشروع في الجدول الأصلي (لأنه جزء من المفتاح الرئيس) .

٣ - نبقي بقية الأعمدة كما هي (عدد الساعات).

٤ - وبالتالي تصبح الجداول على النحو التالي بعد عملية التقسيم :

NO	Project_Code	Hours
210	P1	12
210	p2	20
210	p3	40
201	P1	30
201	p3	15
305	P2	40
305	p3	20

NO	Name
210	Ali
210	Ali
210	Ali
201	salem
201	salem
305	Ali
305	Ali

Project_Code	Deptno	Dname
P1	10	Research
p2	20	Operation
p3	20	Operation

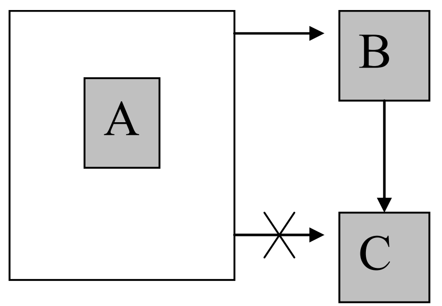
تعريف الصيغة المعيارية الثالثة(Third Normal Form 3NF):

نستطيع أن نقول أن الجدول في الصيغة المعيارية الثالثة:

١ - إذا كان الجدول فيالصيغة المعيارية الثانية.

٢ - إذا لم يحتوي الجدول على اعتمادية متعدية.

الاعتمادية المتعدية: هي أن تعتمد بعض الأعمدة (الصفات) اعتمادا وظيفيا على صفة غير المفتاح الرئيس.



نلاحظ أن A تحدد B, C أي إنB, C تعتمد اعتمادا وظيفيا على A وكذلك أن B تحدد C أي إن C تعتمد اعتمادا وظيفيا B. وفي حالة نستطيع أن نقول أن هذا الجدول يحتوي على اعتمادية متعدية.

والآن هل الجداول السابقة في الصيغة المعيارية الثالثة ؟

وللإجابة على ذلك نجيب على السؤالين التالين:

١ - هل الجداول في الصيغة المعيارية الثانية ؟

نلاحظ أن جميع الجداول في الصيغة المعيارية الثانية حيث لا يوجد فيها اعتمادية جزئية .

٢ - هل توجد هناك اعتمادية متعدية ؟

ولمعرفة ذلك يجب أن نحدد الاعتمادية الوظيفية لكل جدول

أ - الجدول الأول

FD 1 :No → Name

لا توجد اعتمادية متعدية .

ب - الجدول الثاني

FD 1 :No, Project_Code→ hours

لا توجد اعتمادية متعدية.

ج - الجدول الثالث

FD 1 : Project_Code→ Deptno,Dname

FD 2 : Deptno → Dname

المفتاح الرئيس هو Project_Code يحدد Dname وDeptno وفي نفس الوقت فإن وDeptno يحدد Dname أي إن هناك اعتمادية متعدية . وللتخلص من هذه المشكلة يجب أن نقوم بتقسيم الجدول إلى جداول بحيث يضم كل منها الأعمدة التي تعتمد على بعض ونبقي المفتاح مع الأعمدة التي تعتمد عليه وحدة فقط مع إبقاء المحدد الجديد (Deptno)

١ - نقوم بنقل رقم و اسم القسم إلى جدول جديد ونبقي نسخة من رقم القسم في الجدول الأصلي.

٢ - وبالتالي تصبح الجداول على النحو التالي بعد عملية التقسيم :

NO	Name
210	Ali
210	Ali
210	Ali
201	Salem
201	Salem
305	Ali
305	Ali

NO	Project_Code	Hours
210	P1	12
210	p2	20
210	p3	40
201	P1	30
201	p3	15
305	P2	40
305	p3	20

Deptno	Dname

Project_Code	Deptno

P1	10
p2	20
p3	20

10	Research
20	Operation

الآن نستطيع أن نقول أن هذه الجداول هي في الصيغة المعيارية الثالثة **3NF** وتعتبر هذه الصيغة مقبولة لمعظم مصممي قواعد البيانات .

تمارين

١. وضح المقصود بمشاكل تكرار البيانات(Data Anomalies) مع الأمثلة .

٢. ما هي الاعتمادية الوظيفية (Functional Dependency FD)؟

٣. اذكر قواعد الاستنتاج مع ذكر مثال من قاعدة بيانات تسجيل الطلاب على كل قاعدة .

٤. متى يكون الجدول في
أ - الصيغة المعيارية الأولى.
ب - الصيغة المعيارية الثانية .
ج - الصيغة المعيارية الثالثة .

٥. أعط مثالاً على كل مما يلي :
أ - جدول ليس في الصيغة المعيارية الأولى.
ب - جدول ليس في الصيغة المعيارية الثانية .
ج - جدول ليس في الصيغة المعيارية الثالثة .

٦. هل الجدول التالي في الصيغة المعيارية الثالثة ؟ إذا لم يكن كذلك قم بتحويله إلى الصيغة المعيارية الثالثة عل شكل خطوات مع الرسم .

Course _No	Sec_ No	Dept	Credit _Hours	Course Level	Ins_id	Semester	Year	Date	Room _No	No_of _stu
رقم المقرر	الشعبة	القسم	الساعات المعتمدة	مستوى المقرر	رقم المدرس	الفصل الدراسي	السنة الدراسية	وقت الشعبة	القاعة الدراسية	عدد طلاب



تصميم قواعد البيانات

تحويل نموذج الكيانات و العلاقات إلى نموذج علائقي

تحويل نموذج الكيانات و العلاقات إلى نموذج علائقي

٥

الجدارة:

القدرة على تحويل نموذج الكيانات و العلاقات إلى نموذج علائقي.

الأهداف:

أن يستطيع المتدرب تحويل نموذج الكيانات و العلاقات إلى نموذج علائقي.

مستوى الأداء المطلوب:

أن يتقن المتدرب عملية تحويل نموذج الكيانات والعلاقات إلى نموذج علائقي بنسبة ١٠٠٪.

الوقت المتوقع للتدريب:

ساعتان

الوسائل المساعدة:

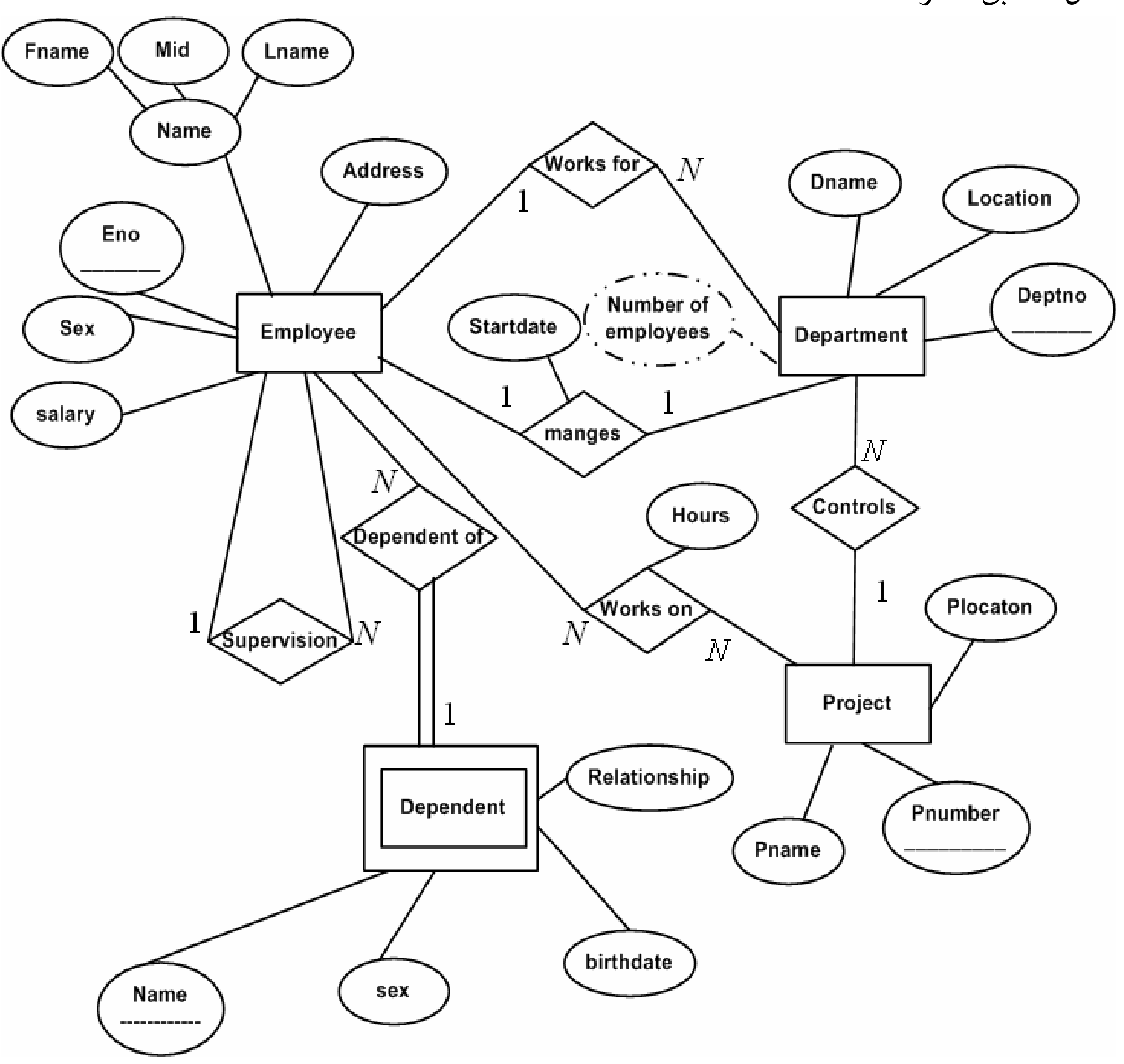
قلم + دفتر

متطلبات الجدارة:

أن يكون المتدرب قد أتقن الجدارة في الوحدات السابقة .

مقدمة

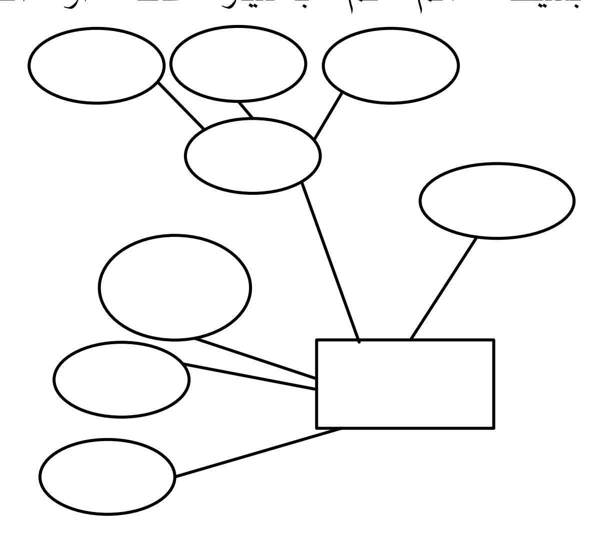
لتحويل عملية التصميم إلى قاعدة بيانات لابد في البداية من تحويل نموذج الكيانات والعلاقات) إلى نموذج علائقي حتى نسهل عملية تنفيذ هذا النموذج في قاعدة (إنشاء الجداول).وسنقوم في هذا الفصل بدراسة كيفية تحول نموذج المفاهيم(نموذج الكيانات والعلاقات) إلى نموذج علائقي مستخدمين المثال السابق للشركة .



والآن سنقوم بعدة خطوات لتحويل نموذج المفاهيم(نموذج الكيانات والعلاقات) إلى نموذج علائقي :

تحويل لكيانات :

١ - لكل كيان(Entity (E في النموذج قم بإنشاء علاقة Relation (R) بحيث تحتوي العلاقة على جميع الصفات البسيطة غير المركبة وإذا كانت الصفات مركبة قم بتقسيمها إلى صفات بسيطة ،ثم قم باختيار صفة أو أكثر لتشكل المفتاح الرئيس للعلاقة .

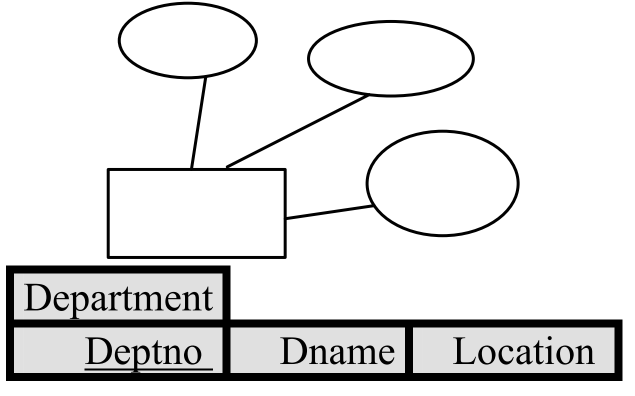


نقوم بتحويلها لتصبح على الشكل التالي:

Employee

Eno	Fname	Mid	Lname	sex	Birthdate	Salary

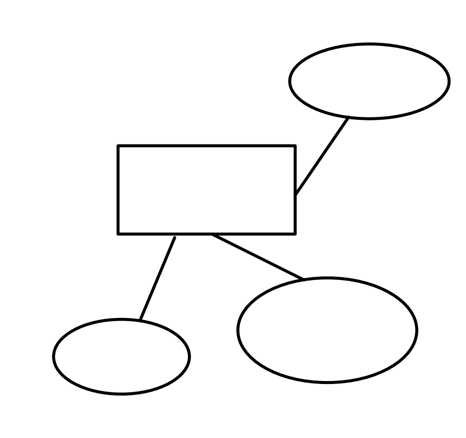
لاحظ أننا قمنا بتقسيم الاسم (صفة مركبة) إلى مكونات بسيطة .

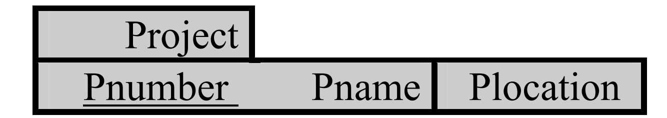


Department

Deptno	Dname	Location

لاحظ أننا لم نقم بإضافة عدد الموظفين (صفة مشتقة) ولكن يجب أن تأخذ بعين الاعتبار لإيجاد عدد الموظفين عن طريق بناء آلية استرجاع (Query).

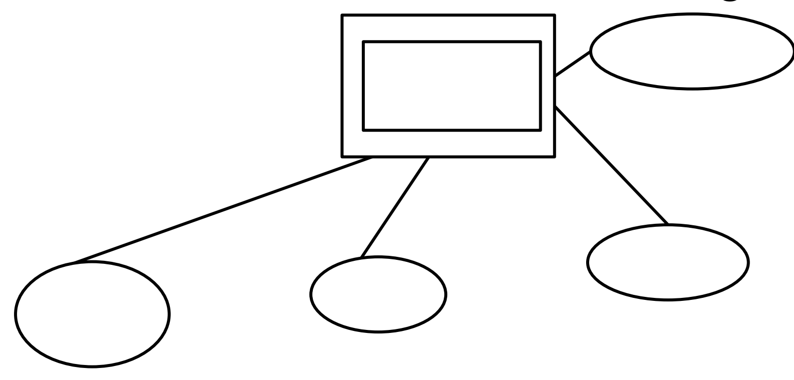




Project		
<u>Pnumber</u>	Pname	Plocation

تحويل لكيانات الضعيفة Weak Entity :

لكل كيان ضعيف (Weak Entity) في النموذج قم بإنشاء علاقة Relation (R) بحيث تحتوي العلاقة على جميع الصفات البسيطة غير المركبة وإذا كانت الصفات مركبة قم بتقسيمها إلى صفات بسيطة ،ثم قم باختيار إحدى الصفات مع المفتاح الرئيس للكيان الذي يتبع إليه الكيان الضعيف لتشكل المفتاح الرئيس للكيان،ثم قم بإنشاء مفتاح أجنبي ليشير إلى الكيان الذي يتبع الكيان الضعيف (المفتاح الرئيس لذلك الكيان).

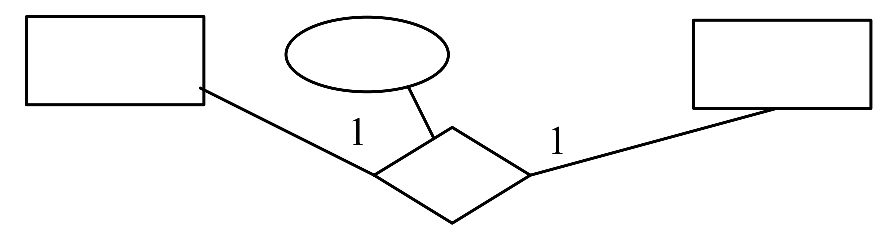


Dependent				
<u>Eno</u>	<u>Name</u>	Sex	Birthdate	Relationship

تحويل التشاركية: كما مر معنا سابقا فهناك ثلاثة أنواع من التشاركية علاقة واحد ـ واحد (١:١) وعلاقة واحد ـ متعدد (N:١) علاقة متعدد ـ متعدد (N:N) وسنقوم بعملية التحويل كل منها على النحو التالي :

١ - علاقة واحد ـ واحد (١:١)

لكل علاقة واحد ـ واحد (١:١) قم باختيار أحد الكيانين لتحتوي على مفتاح أجنبي ليشير إلى الكيان الآخر .

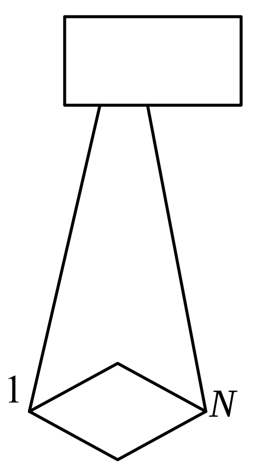


ففي هذه الحالة نقوم بإضافة صفة جديدة (Mgr) لتشير إلى الموظف الذي يتولى إدارة القسم (مفتاح أجنبي لجدول الموظفين) وكذلك إضافة تاريخ بداية إدارة هذا الموظف لذلك القسم .

Department				
<u>Deptno</u>	Dname	Location	Mgr	Startdate

٢ - علاقة واحد ـ متعدد (N: 1)

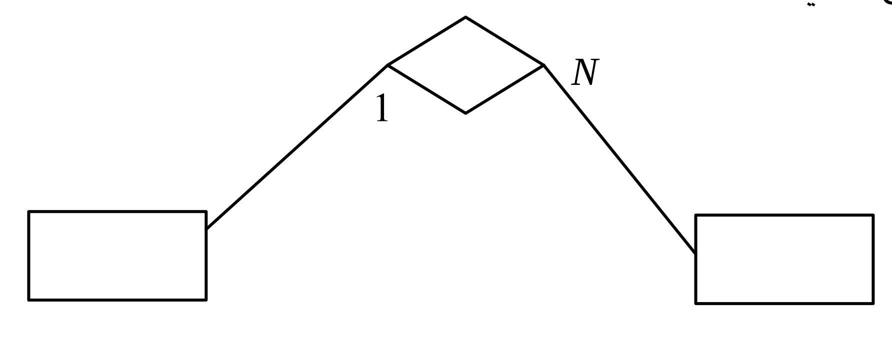
لكل علاقة واحد ـ متعدد(N:1) قم بإضافة عمود(أعمدة) لتكون مفتاحا أجنبيا في جانب المتعدد(N) ليشير إلى المفتاح الرئيس في جانب الواحد(١) .



Employee								
<u>Eno</u>	Fname	Mid	Lname	sex	Birthdate	Salary	Mgr	

Employee

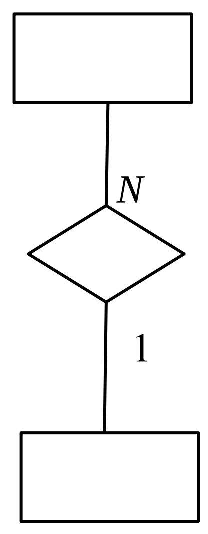
وفي هذه الحالة نقوم بإضافة صفة جديدة (Mgr) لتشير إلى الموظف الذي يتولى الإشراف على الموظف (مفتاح أجنبي لنفس الجدول)



Employee

Eno	Fname	Mid	Lname	sex	Birthdate	Salary	Mgr	Deptno

وفي هذه الحالة نقوم بإضافة صفة جديدة (Deptno) لتشير إلى القسم الذي يتبع إليه الموظف (مفتاح أجنبي لجدول الأقسام)



Project

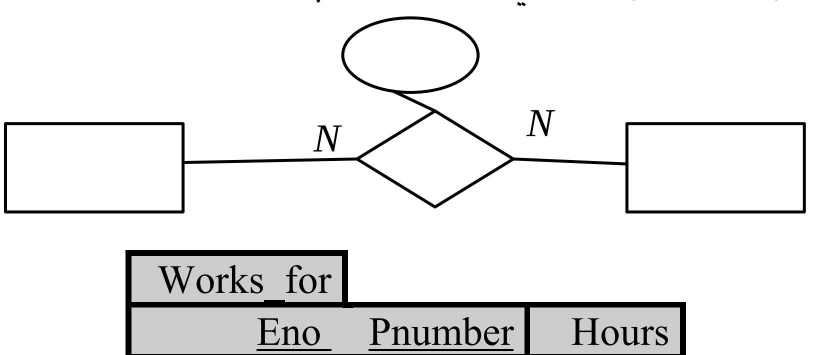
Pnumber	Pname	Plocation	Deptno

وفي هذه الحالة نقوم بإضافة صفة جديدة (Deptno) لتشير إلى القسم الذي يدير هذا المشروع (مفتاح أجنبي لجدول الأقسام).

Employee

٣ -علاقة متعدد ـ متعدد (N:N)

لكل علاقة متعدد ـ متعدد (N:N) قم بإنشاء علاقة جديد يكون المفتاح الرئيس لها عبارة عن دمج المفاتيح الرئيسة في طرفي العلاقة.وإضافة أي صفات جديد لهذه العلاقة



Works_for		
Eno	Pnumber	Hours

ففي هذه الحالة نقوم بإنشاء جدول جديد يحتوي (رمز المشروع،رقم الموظف ،عدد ساعات العمل) بحيث يشكل (رمز المشروع،رقم الموظف) المفتاح الرئيس للجدول وبنفس الوقت يكون رمز المشروع مفتاحاً أجنبياً لجدول المشاريع ، و رقم الموظف مفتاحاً أجنبياً لجدول الموظفين .

تحويل العلاقة بين الأنواع الفرعية(Subtype) والأنواع العليا(Super Type) ISA

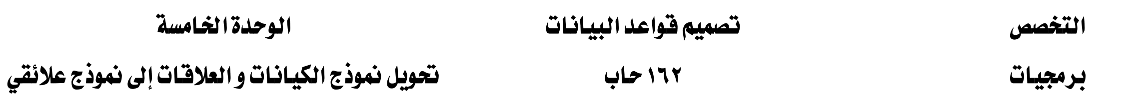
وذلك عن طريق وضع المفتاح الرئيس في النوع الفرعي ليكون مفتاحا رئيسيا لهذا الجدول وفي نفس الوقت يكون مفتاحا أجنبيا للنوع الأعلى:

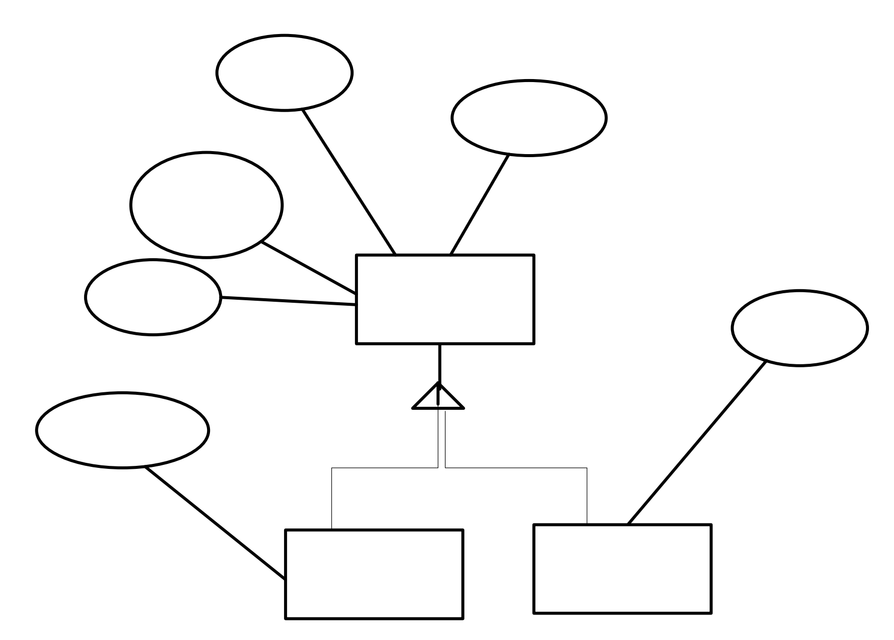
لنفرض أن لدينا نوعين من الموظفين

١ - موظف دائم يكون له راتب ثابت
٢ - موظف يعمل بالساعة ونسجل له أجرة العمل عن كل ساعة

فبالتالي يكون النموذج على الشكل التالي .

Employee





فناتج عملية التحويل يكون على النحو التالي:

S_Employee	
Eno	Salary

H_Employee	
Eno	Hour_Rate_

Eno

Sex

Hour_rate

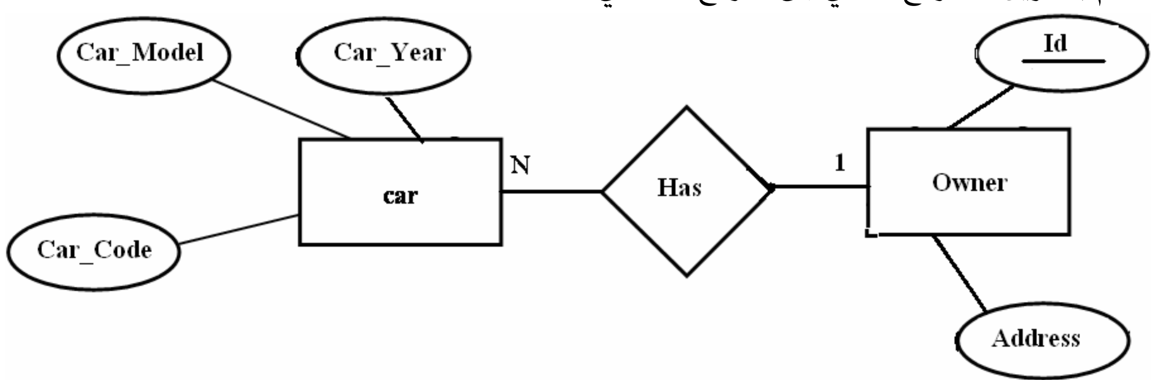
تمارين

١. ما الفائدة من تحويل نموذج الكيانات و العلاقات إلى نموذج علائقي؟

٢. بين باستخدام الرسم كيفية تحويل تشاركية N:N .

٣. بين باستخدام الرسم كيفية تحويل العلاقة بين الأنواع الفرعية(Subtype) والأنواع العليا(Super Type) ISA.

٤. قم بتحويل النموذج التالي إلى نموذج علائقي



٥. قم بتحويل النموذج الناتج من حل السؤال ٦ في الوحدة الثالثة إلى نموذج علائقي.



تصميم قواعد البيانات

تعريف المتغيرات

تعريف المتغيرات

٦

الجدارة:

تعريف واستخدام المتغيرات PLSQL.

الأهداف:

- أن يتعرف المتدرب على تركيب وحدات PLSQL
- أن يتعرف المتدرب على أنواع وحدات PLSQL
- أن يتعرف المتدرب على كيفية تعريف واستخدام المتغيرات بمختلف أنواعها .
- أن يميز المتدرب بين أنواع البيانات Datatypes المختلفة .

مستوى الأداء المطلوب:

أن يتقن المتدرب تعريف واستخدام المتغيرات بنسبة ١٠٠٪.

الوقت المتوقع للتدريب:

ساعتان

الوسائل المساعدة:

- معمل حاسب آلي
- قلم + دفتر

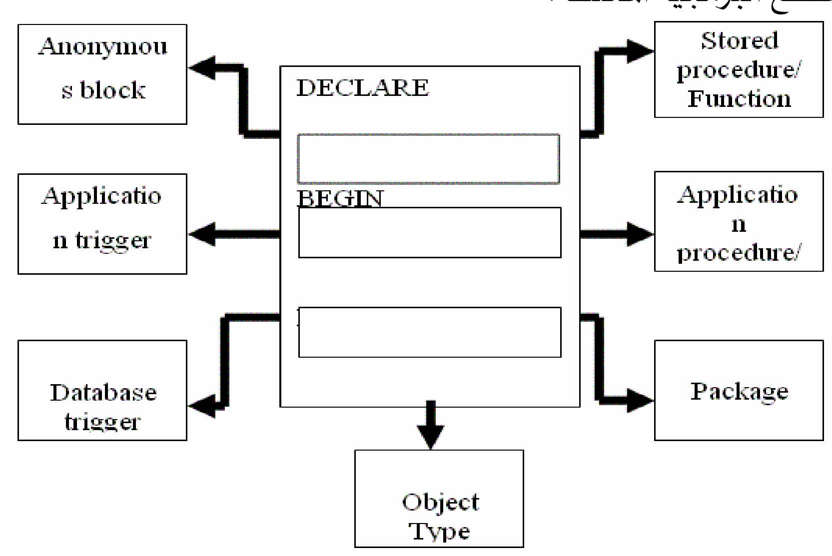
متطلبات الجدارة:

أن يكون المتدرب قد استخدام لغة SQL التي درسها في المقرر السابق لهذا المقرر.

مقدمة:

ما هي لغة PL/SQL ؟ هي عبارة عن تطوير للغة الاسترجاع SQL (Programming Language Structured Query Language) حيث قامت شركة Oracle بعمل هذا التطوير لإعطاء لغة الاسترجاع المزايا اللازمة لمواكبة متطلبات البرمجة مثل كتابة مجموعة من الجمل التي تقوم بحل مسألة معينة بخلاف SQL التي تستخدم جملة واحدة فقط .وكذلك استخدام والمتغيرات وجمل الدوران و الشرط ... إلخ .

وسنتعلم في هذه الوحدة على كيفية تعريف واستخدام والمتغيرات داخل الوحدات البرمجية Modules، في لغة PL/SQL. وقبل ذلك لابد أن نتعرف على هذه الوحدات وتركيبها. إن القطع البرمجية طريقه لبناء البرنامج من مجموعات منفصلة من الوحدات البرمجية Modules ، كل منها يقوم بعمل وظيفة معينة أو مهمة محددة باتجاه الوصول إلى الهدف النهائي في البرنامج، وعندما يتم الانتهاء من كتابة الوحدات البرمجية يمكن تنفيذها مباشرة أو تخزينها في خادم قاعدة البيانات Database Server لتصبح هذه الوحدات كائنات في قاعدة البيانات بحيث يمكن استخدامها من قبل أي وحدة برمجية في قاعدة البيانات هذه. ولتخزين الوحدات البرمجية في قاعدة البيانات يجب إرسال البرنامج المصدري Source Code إلى خادم قاعدة البيانات ليتم ترجمته Compile إلى لغة انتقالية تسمى P-Code. ويوضح الرسم أدناه الأنواع المختلفة للقطع البرمجية المختلفة:



وبشكل عام فإنه يمكننا القول أن الوحدات البرمجية يمكن تقسيمها إلى قسمين أساسيين هما:

١. وحدة برمجية غير مسماة Anonymous Block: وهي الوحدات البرمجية التي ليس لها اسم محدد. ولا يمكن تخزينها في قاعدة البيانات ولكن يتم تحميلها في الذاكرة وتنفيذها عند الحاجة لها. وفي الفصول القادمة سيكون تعاملنا مع هذا النوع فقط من الوحدات.

٢. وحده برمجية معروفة Named Block: وتسمى أحيانا Subprograms، وهي الوحدات البرمجية التي لها اسم محدد عند تعريفها ويندرج ضمنها: Function, Procedure, Trigger, Package كلها لها أسماء محددة.

ويمكن للقطعة البرمجية إن تحتوي على وحدة برمجية واحدة أو أكثر وبالتالي يمكن أن توجد الوحدات البرمجية بداخل بعضها Nested Blocks.

تركيب الوحدات (Blocks) :

تتكون الوحدة (Block) من ثلاثة أجزاء:

١. **جزء الإعلان (التصريح)** Declarative وفيه يتم تعريف المتغيرات التي سيتم استخدامها في هذه الوحدة (Block)، كذلك تعريف المؤشرات (Cursors) والاستثناءات المعرفة من قِبل المستخدم وستتم عملية التعرف على المؤشرات والاستثناءات في الفصول الخاصة بذلك وهذا الجزء اختياري أي يمكن كتابة وتنفيذ وحدة (Block) لا تحتوي على متغيرات .

٢. **الجزء التنفيذي:Executable** ويحتوي على جملSQL التي تقوم بالتعامل مع البيانات الموجودة في قاعدة البيانات مثل (الاسترجاع الإضافة، التعديل، الحذف) ويحتوي كذلك على جمل PLSQL والتي تقوم بالتعامل مع البيانات في الوحدة (Block) مثل الإدخال، الإخراج، الدوران.....إلخ وهذا الجزء إجباري لأنه يحتوي على الجمل الواجب تنفيذها.

٣. **الجزء الخاص بمعالجة الاستثناءات (Exception)**: وفي هذا الجزء تتم معالجة الأخطاء المحتمل حدوثها خلال مرحلة التنفيذ عن طريقة بيان الإجراء اللازم عمله عند حدوث مثل هذه الأخطاء.

DECLARE	→	اختياري تعريف المتغيرات، المؤشرات واستثناءات المستخدم
BEGIN	→	إجباري بداية الجز ء التنفيذي
▪ SQL	→	جمل SQL
▪ PLSQL	→	جمل PLSQL
EXCEPTION	→	اختياري الإجراء الذي يجب تنفيذه عند حدوث خطأ(استثناء)
END;		إجباري نهاية الجزء التنفيذي

استخدام المتغيرات:

المتغيرات هي عبارة عن مواقع في الذاكرة يمكن استخدامها للتخزين المؤقت للبيانات (خلال عملية تنفيذ وحدة (Block) PLSQL :

فوائد استخدام المتغيرات:

١. **معالجة البيانات المخزنة:**

يمكن استخدام المتغيرات لتحتوي على القيم المخزنة في قاعدة البيانات وبالتالي يمكن استخدامها في العمليات الحسابية دون الحاجة إلى الرجوع إلى قاعدة البيانات.

٢. **إعادة الاستخدام:**

عند تعريف المتغير يتم حجز مكان لهذا المتغير في الذاكرة وبالتالي يمكن تخزين واسترجاع البيانات في ومن هذا المكان أكثر من مرة خلال عملية تنفيذ البرنامج .

٣. **سهولة الصيانة:**

عند استخدام **Type%** و**ROWTYPE%** (سيتم شرحها في لاحقا في هذا الفصل) نقوم بعملية تعريف متغير بناءً على تعريف متغير آخر أو مؤشر أو عمود في قاعدة البيانات وبالتالي في حالة تغيير تعريف العمود أو المؤشر فلا تلزم عملية إعادة التعريف لهذا المتغير وهذا يوفر عملية التعديل محافظا على التوافقية مع قاعدة البيانات .

تعريف المتغيرات:

لاستخدام المتغيرات في وحدات PLSQL لابد من تعريف هذه المتغيرات قبل عملية استخدامها وخلال عملية التعريف هناك إمكانية إسناد قيم ابتدائية لهذه المتغيرات،ويجب التنبه إلى وضع جملة تعريف منفصلة لكل متغير نرغب بتعريفه وكذلك مراعاة أن كل جملة في وحدة (Block) PLSQL يجب أن تنتهي بفاصلة منقوطة (;) . وهذا الشكل العام لجملة تعريف المتغيرات:

```
identifier [CONSTANT] datatype [NOT NULL] [:= DEFAULT | expression ] ;
```

Identifier	اسم المتغير
CONSTANT	وهذا قيد على المتغير بحيث تمنع عملية تعديل قيمة هذا المتغير بعد عملية إسناد قيمة له. إذا عرفنا المتغير باستخدام CONSTANT فلابد من وضع قيمة ابتدائية له.
datatype	نوع البيانات التي يمكن تخزينها في المتغير
NOT NULL	يجب أن يحتوي على قيم، والمتغيرات المعرفة NOT NULL يجب وضع قيم ابتدائية لها.
expression	أي تعبير مقبول PLSQL فيمكن أن يكون قيمة، أو متغير آخر أو تعبير يحتوي على قيم ومتغيرات وعمليات (حسابية أو غيرها)

مثال :

Declare	
v_name VARCHAR2(10) ;	متغير يحتوي على بيانات من نوع السلاسل الرمزية متغيرة الطول بطول عشرة رموز
v_date DATE ;	متغير يحتوي على بيانات نوع تاريخ
v_id NUMBER(2) NOT NULL :=10 ;	متغير يحتوي على بيانات رقمية بطول عشرة خانات وبقيمة ابتدائية ١٠ وبشرط أن لا يكون NULL
V_comm CONSTANT NUMBER :=120;	متغير يحتوي على بيانات رقمية بشرط أن لا تتغير قيم هذا المتغير خلال عملية تنفيذ البرنامج وبقيمة ابتدائية ١٢٠
BEGIN …… END;	

النقاط التي يجب مراعاتها خلال عملية تعريف المتغيرات :

١. اتباع قواعد التسمية المستخدمة في SQL .

- لا يمكن أن يكون هناك أكثر من كائن يحمل نفس الاسم إلا إذا كان في وحدة (Block) أخرى.
- يجب عدم استخدام اسم جدول أو عمود سيتم استخدامها في نفس الوحدة (Block) .
- أن لا يزيد طول الاسم عن ٣٠ حرف .
- أن يحتوي على الرموز التي يمكن استخدامها في التسمية في SQL .
- A-Z, a-z, 0-9, _, #,$

٢. وضع قيم ابتدائية للمتغيرات المعرفة باستخدام NOT NULL وCONSTANT فعند عدم وضع قيم ابتدائية ستحصل على خطأ في التعريف .

٣. وضع قيم ابتدائية باستخدام=: أو باستخدام الكلمة المحجوزة DEFUALT .

٤. تعريف متغير واحد فقط في كل جملة .

إسناد القيم للمتغيرات:

هناك طريقتان لعملية إسناد القيم للمتغيرات

- باستخدام جملة الإسناد وفيها يتم كتابة اسم المتغير متبوعا ب **=:** ثم وضع التعبير .

Identifier := expression ;

مثال :

في هذه الجملة يتم إسناد القيمة Ali للمتغير v _name	v_name :='Ali' ;
في هذه الجملة يتم إسناد ناتج عملية حسابية للمتغير net_sal	net_sal := v_sal – v_sal*.08;

- وهناك طريقة أخرى وهي إسناد القيم خلال جملة الاسترجاع من قاعدة البيانات **SELECT**
- مثال

```
SELECT    ename
INTO      v_name
FROM      emp
WHERE     empno = 7788;
```

في هذه الجملة تتم عملية إسناد القيمة الراجعة للعمود ename من عملية الاسترجاع للمتغير v_name

ولكن في هذه الحالة يجب أن نتأكد من أن عملية الاسترجاع تعيد قيمة واحدة فقط للعمود ename وإلا سيؤدي ذلك إلى حدوث خطأ .

أنواع البيانات للمتغيرات Datatypes :

١ - المتغيرات التي تحتوي على قيمة واحدة Scalar Datatype:

وهذا النوع من المتغيرات يمكن أن يحتوي على قيم مفردة. والجدول التالي يمثل وصفاً لهذه الأنواع:

النوع	الوصف
VARCHAR2(size)	البيانات الرمزية متغيرة الطول ويمثلsize أكبر عدد من الرموز التي يمكن تخزينها في المتغير.ويجب تحديد الطول عند عملية التعريف . أكبر حجم هو ٣٢,٦٧٦ Byte
CHAR[(SIZE)]	البيانات الرمزية ثابتة الطول ويمثلsize أكبر عدد من الرموز التي يمكن تخزينها في المتغير .وإذا لم يتم تحديد الطول تكون القيمة الافتراضية له ١ أكبر حجم هو ٣٢,٦٧٦ Byte
NUMBER (precision,scale)	البيانات الصحيحة والكسرية ويمثل Precision الحجم الكلي للمتغير و scale يمثل عدد المنازل العشرية
DATE	ويمثل نوع البيانات التي تكون على شكل تاريخ (وقت و تاريخ) والقيم التي يمكن أن يحتويها ما بين ٤٧١٢ قبل الميلاد و ٩٩٩٩ بعد الميلاد .
LONG	البيانات الرمزية متغيرة الطول ويمثلsize أكبر عدد من الرموز التي يمكن تخزينها في المتغير.ويجب تحديد الطول عند عملية التعريف . أكبر حجم هو ٣٢,٦٧٠ Byte وأكبر حجم للعمود في الجدول من نوع LONG هو ٢,١٤٧,٤٨٣,٦٤٧Byte
LONG RAW	البيانات الممثلة ثنائيا (Binary) مثل الصور .
BOOLEAN	البيانات المنطقية مثل TRUE,FLASE
BINARY_INTEGER	أعداد صحيحة بين ٢,١٤٧,٤٨٣,٦٤٧ -و٢,١٤٧,٤٨٣,٦٤٧
PLS_INTEGR	أعداد صحيحة بين٢,١٤٧,٤٨٣,٦٤٧ -و٢,١٤٧,٤٨٣,٦٤٧ ولكن بحجم أقل من NUMBER و BINARY_INTEGER

مثال:

```
v_job             VARCHAR2(9);
v_count           BINARY_INTEGER := 0;
v_total_sal       NUMBER(9,2) := 0;
v_orderdate       DATE := SYSDATE + 7;
c_tax_rate        CONSTANT NUMBER(3,2) := 8.25;
v_valid           BOOLEAN NOT NULL := TRUE;
```

استخدام الخاصية TYPE% في التعريف وتستخدم لتعريف متغير بالاعتماد على تعريف متغير آخر أو تعريف عمود في جدول في قاعدة البيانات .

مثال

```
v_ename        emp.ename%TYPE;
```

تعريف المتغيرv_ename بنفس النوع والحجم للعمودename الموجود في جدولemp

```
v_balance        NUMBER(7,2);
```

تعريف المتغيرv_bqlqnce من نوع رقمي

```
v_min_balance        v_balance%TYPE := 10;
```

تعريف المتغير v_min_balance بنفس النوع والحجم للمتغيرv_balance وبقيمة ابتدائية ١٠.

تعريف المتغيرات المنطقية BOOLEAN :

-القيم لهذه المتغيرات هي فقط TRUE , FLASE, NULL .

- يمكن ربط المتغيرات بواسطة العمليات المنطقية AND,OR ,NOT

يمكن استخدام التعبيرات الحسابية والرمزية و تعبيرات الوقت للحصول على نتائج منطقية (A<B) ستعيد إماTRUE أو FALSE .

مثال:

```
V_Sal1 NUMBER: =1000 ;
V_Sal2 NUMBER: =2500 ;
 Valid    BOOLEAN:= (V_Sal1          >
 V_Sal2);
```

تعريف المتغير Valid بحيث يحتوي على ناتج عملية مقارنة V_SAL1 > V_SAL2 والتي ستكون في هذه الحالة TRUE أي إن القيمة الابتدائية ل Valid ستكون TRUE .

٢ - **المتغيرات المركبة** Composite Datatype :

وهي على نوعين

أ - **السجلات** RECORDS وتتكون من عدة حقول ولا يشترط أن تكون هذه الحقول من نفس النوع أو الحجم .

مثال : يمكن تعرف السجل EMP_REC بحيث يحتوي على اسم ورقم وراتب الموظف

EMP_REC		
SALARY	NAME	ID

3000	ALI	١٠١

ب -**الجداول** TABLES وتشبه السجلات في أنها تتكون من عدة حقول ولا يشترط أن تكون هذه الحقول من نفس النوع ولكن يمكن أن يحتوي الجدول على أكثر من صف (تعتبر مصفوفة من السجلات) .وسيتم بحث هذه الأنواع في فصول أخرى .

٣ - **المتغيرات التي تحتوي على كائنات كبيرة الحجم** LOB Datatype (Large Objectas) :

ويمكن أن تحتوي هذه المتغيرات على كائنات كبيرة الحجم مثل الأفلام والصور والنصوص كبيرة الحجم. وبحث هذه الأنواع خارج نطاق هذا الكتاب .

٤ - **متغيرات الربط Bind Variables** : وهي المتغيرات التي يتم تعريفها داخل البيئة التي يتم تنفيذ الوحدة (Block) داخلها مثل (SQL* Plus) ويمكن استخدام هذه المتغيرات داخل وحدة (Block) أو أكثر وقراءة وتخزين قيم داخلها خلال عملية تنفيذ الوحدة (Block) .

وتتم عملية تعريف متغيرات الربط كما يلي:

تعريف متغير ربط Emp_Sal من نوع NUMBER على مستوى المضيف (Host)	SQL > VARIABLE Emp_Sal NUMBER
تعريف متغير ربط Emp_ Name من نوع VARCHAR2(2)بطول ٢٠ حرف على مستوى المضيف (Host) .	SQL > VARIABLE emp_name varchar2(20)

وتستخدم متغيرات الربط داخل الوحدة (Block) كما تستخدم بقية المتغيرات المعرفة داخل الوحدة (Block) ولكن تسبق متغيرات الربط بنقطتين علويتين قبل اسم المتغير (:)

: Emp_Name:= ‘AHMED’;

ويمكن طباعة متغير الربط في بيئة SQL* Plus باستخدام الأمر PRINT

```
SQL> PRINT   Emp_ Name
```

جملة الإخراج:

تتم عملية طباعة المخرجات من وحدة (Block) على الشاشة وذلك باستخدام الإجراء PUT_LINE الموجود في الحزمة DBMS_OUTPUT وذلك على الشكل التالي :

DBMS_OUTPUT.PUT_LINE ('Well Come to PLSQL Programming'**);**

ولرؤية النتائج لابد أن تمكن الحزمة من العمل وذلك باستخدام الأمر

```
SQL >SET SERVEROUTPUT ON
```

ويمكن أيضا كتابة هذا الأمر في بداية الوحدة (Block) .

```
SET SERVEROUTPUT ON
 ACCEPT p_annual_sal PROMPT ' Please enter the annual salary '
 Declare
 V_sal  NUMBER(9,2) :=&p_annual_sal ;
 BEGIN
 V_sal  :=V_sal  /12;
 DBMS_OUTPUT.PUT_LINE('Monthly Salary is '||TO_CHAR(V_sal));
 END;
```

تمارين

١. أي من الجمل التالية تعتبر جمل صحيحة لتعريف المتغيرات في PLSQL وأيها خطأ ولماذا ؟

1-	DECLARE VI_ID NUMBER (4);

2-	DECLARE v_x ,v_y ,v_z VARACHAR2(10);

3-	DECLARE V_Date DATE NOT NULL;

4-	DECLARE V_valid BOOLEAN :=1;

٢. حدد نوع البيانات الناتج عن تنفيذ كل من الجمل التالية

1-	V_days := v_date-SYSDATE;

2-	V_sender := USER||' :'|| TO_CHAR(V_DEPTNO);

3-	v_sum := $100 + $3000;

4-	V_days:=v_date-SYSDATE;

5-	v_flage := TRUE;

6-	v_n1 := v_n2 > (2 * v_n3);

7-	v_value := NULL;

٣. اكتب وحدة (Block) PLSQL لطباعة MY PLSQL WORKS على الشاشة .

```
G_MESSAGE
--------------------------------
My PL/SQL Block Works
```

٤. قم بكتابة وحدة (Block) PLSQL بحيث تحتوي على متغيرين V-CHRوقيمته
’42 is the answer’
و V_NUM وقيمته أول حرفين من V-CHR ثم قم بوضع قيم كل من المتغيرين في متغير ربط ومن ثم قم بطباعة متغيرات الربط من خلال SQL * Plus قم بتخزين الوحدة Block في ملف p6q4.sql .

```
SQL> PRINT g_char

G_CHAR
----------------------
42 is the answer

SQL> PRINT g_num

    G_NUM
----------
      42
```

المملكة العربية السعودية
المؤسسة العامة للتعليم الفني والتدريب المهني
الإدارة العامة لتصميم وتطوير المناهج



تصميم قواعد البيانات

كتابة الجمل التنفيذية

كتابة الجمل التنفيذية

٧

الجدارة:

القدرة على كتابة الجمل التنفيذية داخل الوحدة .

الأهداف:

- أن يتعرف المتدرب على مزايا الجزء التنفيذي من الوحدة (Block).
- أن يقوم المتدرب بكتابة الجمل التنفيذية.
- أن يتعرف المتدرب على قواعد استخدام الوحدات المتداخلة.
- أن يقوم المتدرب بكتابة وتنفيذ الوحدات .
- أن يستخدم المتدرب قواعد تسمية المتغيرات.

مستوى الأداء المطلوب:

أن يتقن المتدرب كتابة الجمل التنفيذي بنسبة ١٠٠٪.

الوقت المتوقع للتدريب:

ساعتان

الوسائل المساعدة:

- معمل حاسب آلي.
- قلم + دفتر

متطلبات الجدارة:

أن يكون المتدرب قد أتقن تعريف واستخدام المتغيرات التي درسها في الوحدة السابقة.

مقدمة

سنتعرف في هذا الفصل على كيفية كتابة الجمل التنفيذية داخل وحدة (Block) PLSQL، وكذلك سنتعرف على القواعد المستخدمة لكتابة الجمل والمتغيرات داخل الوحدة,(Block) ومعرفة مجال المتغيرات في الوحدات المتداخلة.

تركيب الجملة (Syntax) في PLSQL

تتكون الجملة PLSQL من مجموعة من الوحدات (مغيرات، وقيم) و يفصل بين هذه الوحدات بأحد الفواصل مثل الفراغات، المحددات، الملاحظات... .وتخضع عملية كتابة الجمل إلى مجموعة من القواعد التي تضبط ترتيب هذه الوحدات والفواصل التي تفصلها.ويمكن أن تقسم الجملة على أكثر من سطر.

- **مكونات الجملة :**

1. **المتغيرات (الأسماء):**

- يجب أن لا يزيد طول الاسم عن ٣٠ رمزاً.
- أن لا تكون إحدى الكلمات المحجوزة
- يجب أن يبدأ بحرف.
- يجب أن لا يكون اسماً لجدول أو عمود سيستخدم في هذه الوحدة (Block) .

أمثلة مقبولة :

```
v_name varchr2(20);
xyx number;
birth_date  date;
```

أمثلة غير مقبولة :

```
1no number ;
Dept%id number(2);
Select varchar2(10);
```

2. **القيم الثابتة Literal Values :**

- القيم الثابتة الرمزية Character و قيم التاريخ Date يجب أن تكون داخل علامتي تنصيص مفردة (' ') .

```
v_name:='ali';
```

- القيم الرقمية Numbers يمكن أن تكون أعداداً صحيحة أو أعداداً كسرية .

```
v_id :=201;
```

٣. العمليات **Operations**:

-الأس والنفي (NOT,**) .

- الجمع والطرح(+، -)

-الضرب والقسمة (*، /)

- عمليات المقارنة (=,>,<,<=, <=, IS NULL, LIKE, BETWEEN, IN).

- - العمليات المنطقية (AND OR)

- **الملحوظات والتعليقات** Comments :

عبارة عن أي جملة تستخدم لتوضيح عمل البرنامج ودلالات المتغيرات فيسهل على من يريد استخدام أو تعديل البرنامج فهم تركيب وعمل هذا البرنامج .

- تعليقات السطر الواحد تكون بوضع (--) في بداية جملة التعليق وتعني أن ما بعد هذه الإشارة هو نص توضيحي وليس تنفيذي .

- تعليقات السطور المتعددة تكون بوضع (*/) في البداية ووضع (/*) وهذا يعني أن ما بين هاتين الإشارتين هو نص توضيحي وليس تنفيذي.

```
.....
V_name varchar2(20) ;-- this variable used to hold the employee name
Begin
/* this code is used to read
The employee salary and calculate the annual salary
And print the annual salary
*/
.......
End ;
```

استخدام الدوال Using Functions .

الدوال التي يمكن استخدامها داخل الوحدة (Block):

١. **دوال الصف الواحد في SQL** :

- الدوال الرقمية Number Functions مثل (ROUND ,TRUNC, SQRT ....)

```
v_sal:= ROUND(v_sal,2) ;
```

- الدوال الرمزية Character Functions (CONCAT, INITCAP, LOWER,...)

```
SELECT INITCAP (enam) INTO v_name
```

```
FROM emp
WHERE empno =7788;
```

- دوال التاريخ Date Functions (ADD_MONTHS, MONTHS_BETWEEN,…)

```
Num_months := MONTHS_BETWEEN(SYSDATE,v_date);
```

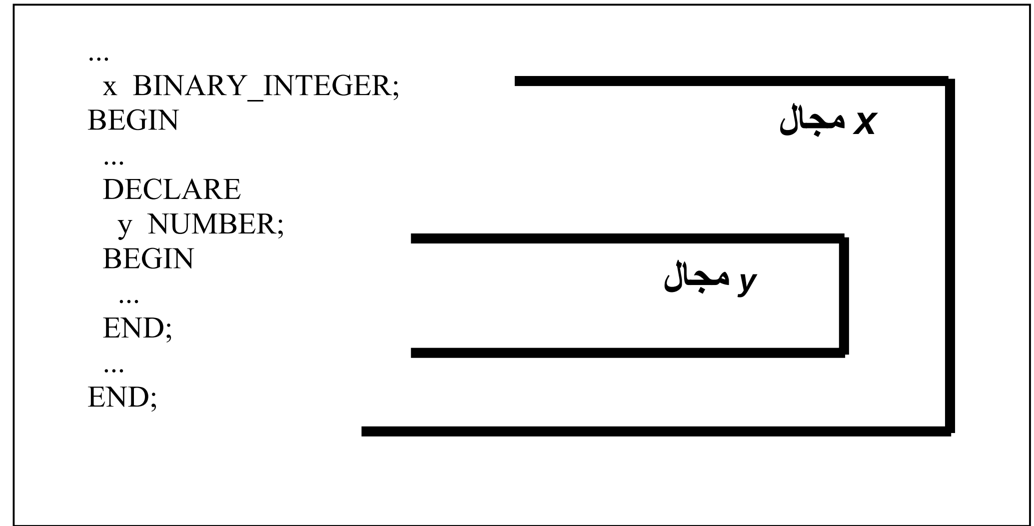
- دوال **SQL** غير مسموح استخدامها
- **Decode**
- **Group Functions** مثل (MIN, MAX, AVG, …..)

٢. دوال التحويل بين أنواع البيانات المختلفة :

- التحويل إلى قيم رمزية CHARACTER ◄ TO_CHAR
- التحويل إلى قيم رقمية NUMBER ◄ TO_NUMBER .
- التحويل إلى قيم تاريخ DATE ◄ TO_DATE .

الوحدات المتداخلة (Nested Blocks)

يمكن كتابة وحدة (Block) داخلية في أي مكان ويمكن كتابة أي جملة تنفيذية، وتعامل الوحدة الداخلية(Nested Block) كجملة تنفيذية ،ويمكن وضع وحدة (Block) في جزء الاستثناءات أيضا .أما بالنسبة لمجال المتغير فتمثل المنطقة التي يمكن التعامل مع المتغير داخلها .



ففي الشكل السابق نلاحظ تعريف **x** على مستوى الوحدة(Block)الخارجية فيكون مجال **x** في الوحدة (Block) التي عرف فيها وكذلك في جميع الوحدات الداخلية التي يمكن أن تعرف داخل هذه الوحدة (Block) . أما بالنسبة ل **y** فيكون معرفاً داخل الوحدة الداخلية والوحدات (Blocks) التي يمكن أن تعرف داخلها ولكنها غير معروفة داخل الوحدة (Block) الخارجية .

- ولكن يجب التنبه إلى أنه في حالة تعريف متغيرين بنفس الاسم في الوحدات المتداخلة فإن الوحدة (Block) تتعامل مع المتغير الأقرب لها .

```
. . .
   x  BINARY_INTEGER;
BEGIN
  ...
 DECLARE
  X     NUMBER;
 BEGIN
   ...
END;
  ...
END;
```

أي استخدام X داخل هذه الوحدة (Block) يكون ل

```
DECLARE
  v_sal            NUMBER(7,2) := 60000;
  v_comm           NUMBER(7,2) := v_sal * .20;
  v_message        VARCHAR2(255) := ' eligible for commission';
BEGIN
```

```
DECLARE
v_sal           NUMBER(7,2) := 50000;
v_comm          NUMBER(7,2) := 0;
v_total_comp    NUMBER(7,2) := v_sal + v_comm;
BEGIN
v_message := 'CLERK not'  ||  v_message;
END;
```

```
v_message := 'SALESMAN'||v_message;
 END;
```

بناء على الشكل السابق حدد قيمة كل من المتغيرات التالية :

- v_message- في الوحدة الداخلية (Sub Block)
- v_total_comp في الوحدة(Block) الرئيسة
- v_comm في الوحدة الداخلية (Sub Block)
- v_comm في الوحدة(Block) الرئيسة
- v_message- في الوحدة(Block) الرئيسة

دليل كتابة البرنامج (Programming Guidelines)

تعتبر عملية صيانة البرنامج من أهم التحديات التي تواجه المبرمج ،ولتسهيل هذه المهمة لابد من أن يكون البرنامج واضحاً وسهل القراءة والتتبع كي يستطيع الشخص الذي سيقوم بعملية الصيانة للبرنامج من فهمه فهما صحيحا . وإليك بعض الأدلة الواجب اتباعها حتى يكون البرنامج واضحا وسهل الفهم:

١. كتابة التعليقات والتوضيحات بحيث تغطي هذه الملحوظات وصف البرنامج وطريقة عمله وكذلك توضيح دلالة المتغيرات وما الذي تعنيه هذه المتغيرات.

٢. استخدام حالة الأحرف الصغيرة والكبير لتكوّن طريقة (Case convention) متعارف عليها لتسمية المتغيرات والكائنات الأخرى ويمثل الجدول التالي بعض الأدلة في التسمية:

الفئة	حالة الأحرف	مثال
جمل SQL	أحرف كبيرة	SELECT, INSERT
الكلمات المحجوزة	أحرف كبيرة	DECLARE ,BEGIN, END
المتغيرات والمعاملات	أحرف صغيرة	v_sal, id, g_sal
أسماء الجداول والأعمدة	أحرف صغيرة	emp, dept, ename

والجدول التالي يبين طريقاً لتسمية المتغيرات حسب أنواع هذه المتغيرات :

الاسم	طريقة التسمية	مثال
المتغيرات variables	v_name	v_empno, v_sal
الثوابت	c_name	c_sal , c_tax
المؤشرات cursors	Name_cursor	emp_cursor,
الاستثناءات exception	e_name	e_too_many
متغيرات الاستبدال substitute variables	p_name	p_empno
المتغيرات العامةglobal	g_name	g_sal

٣. استخدام الإزاحات خلال عملية الكتابة حتى يكون البرنامج واضحا وسهل القراءة :

نلاحظ فرق وضوح الجمل في حالة استخدام إزاحة أو عدم استخدامها .

BEGIN IF x=0 THEN y:=1; ELSE y:=2; END IF; END;	BEGIN IF x=0 THEN y:=1; ELSE y:=2; END IF; END;

تمارين

١ -

```
DECLARE
 v_weight  number(3):=600;
v_message         VARCHAR2(255):='Product 10012';
BEGIN
                /*  SUB BLOCK الوحدة الداخلية */
     DECLARE
      v_weight   number(3):=1;
     v_message  VARCHAR2(255):='Product 11001';
     v_new_loc  VARCHAR2(50):=  'Europe ';
     BEGIN
               v_weight  :=        v_weight  +1;
      v_new_loc:=  'Western '||       v_new_loc;
       END;
     v_weight  :=        v_weight  +1;
     v_message := v_message || 'is in Stock ';
     v_new_loc:=  'Western '||       v_new_loc;
END ;
```

بناء على الشكل السابق حدد قيم ونوع البيانات لكل من المتغيرات حسب قواعد مجال المتغيرات:

أ - قيمة v_weight في الوحدة الداخلية(Sub Block) .

ب - قيمة v_new_loc في الوحدة الداخلية(Sub Block) .

ج - قيمة v_weight في الوحدة الرئيسة (main Block) .

د - قيمة v_message في الوحدة الرئيسة (main Block) .

هـ -قيمة v_new_loc في الوحدة الرئيسة(main Block) .

٢ - قم بكتابة وتنفيذ وحدة (Block) بحيث يقوم المستخدم بإدخال عددين وطباعة حاصل نتيجة قسمة العدد الأول على الثاني مضافا إليها العدد الثاني، وقم بتخزين النتيجة في متغير ثم قم بطباعة هذا المتغير.

مثال

```
Please enter the first number : 2
Please Enter The Second  Number :4

G_RESULT
-----------------------
4.5
```

٣ - قم بكتابة وتنفيذ وحدة (Block) بحيث يقوم المستخدم بإدخال الراتب السنوي للموظف وكذلك إدخال العلاوة . وبعد ذلك حساب إجمالي الراتب وهو عبارة عن الراتب السنوي مضاف إليه حاصل ضرب قيمة العلاوة في الراتب . ثم طباعة الإجمالي.

ملحوظة:

- يجب تحويل العلاوة إلى نسبة مئوية (إذا قام المستخدم بإدخال ١٥ يجب أن تحول إلى **0.15**) .
- إذا لم يقم المستخدم بإدخال قيم للعلاوة يجب أن تعتبر صفراً (استخدم NVL).

```
Please enter the salary amount: 50000
Please enter the bonus percentage: 10

PL/SQL procedure successfully completed.


  G_TOTAL
----------
    55000
```



تصميم قواعد البيانات

التفاعل مع خادم Oracle

التفاعل مع خادم Oracle

٨

الجدارة:

القدرة على التفاعل مع خادم Oracle من داخل الوحدة(Block).

الأهداف:

- أن يقوم المتدرب بكتابة جملة استرجاع ناجحة داخل الوحدة (Block)
- أن يقوم المتدرب بمعالجة البيانات من داخل الوحدة (Block)
- أن يتعرف المتدرب على كيفية التحكم بالعمليات Transactions
- أن يتعرف المتدرب على كيفية استخدام مؤشرSQL Cursor SQL

مستوى الأداء المطلوب:

أن يتقن المتدرب عمليات التفاعل مع خادم Oracle بنسبة ١٠٠٪.

الوقت المتوقع للتدريب:

ساعتان

الوسائل المساعدة:

- معمل حاسب آلي
- قلم + دفتر

متطلبات الجدارة:

أن يكون المتدرب قد أتقن تعريف واستخدام المتغيرات وكتابة الجمل التنفيذية .

مقدمة

إن عملية استرجاع ومعالجة البيانات المخزنة تتطلب استخدام جمل SQL و جمل PL/SQL للقيام بهذه العمليات ولكن يجب ملحوظة أن وحدة PL/SQL ليست عملية كاملة (Transaction) ولكن يمكن استخدام COMMIT, SAVEPOINT, ROLLBACK من داخل الوحدة Block، ويجب معرفة أن جمل DDL يمكن استخدامها داخل الوحدة Block مثل CREATE TABLE, ALTER TABLE وكذلك جمل DCL مثل GRANT, REVOKE .

كتابة جملة الاسترجاع Select Statement

```
SELECT  select_list
INTO  {variable_name[, variable_name]...
      | record_name}
FROM  table
WHERE  condition;
```

select_list	البيانات التي نرغب باسترجاعها من قاعدة البيانات مثل الأعمدة العمليات والتعبيرات الحسابية
INTO	إجبارية. وتعني تخزين Select_List في Variables أو record_name
variable_name	المتغير(المتغيرات) التي سيتم وضع القيم المسترجعة داخلها يجب أن يكون عدد المتغيرات مساوياً لعدد البيانات المسترجعة select_list
record_name	اسم السجل الذي سيتم وضع القيم المسترجعة داخله
table	اسم الجدول
condition	شرط الاسترجاع (يجب أن تعيد جملة الاسترجاع قيمة واحدة فقط لكل عمود وإلا سيؤدي ذلك إلى ظهور استثناء Exception

مثال:

```
DECLARE

  v_deptno    NUMBER(2);
  v_loc       VARCHAR2(15);

BEGIN

  SELECT        deptno, loc
  INTO          v_deptno, v_loc
  FROM          dept
  WHERE         dname = 'SALES';

END;
```

في المثال السابق تتم عملية استرجاع رقم ومكان القسم الذي يحمل الاسم 'SALES' وتخزين هذه القيم في المتغيرات v_deptno و v_loc على الترتيب .

ولكتابة جمل الاسترجاع لابد من مراعاة النقاط التالية :

- يجب أن تنتهي جملة الاسترجاع بفاصلة منقوطة (;).
- يجب أن تحتوي جملة الاسترجاع على **INTO** .
- عدد المتغيرات يجب أن يساوي عدد القيم الراجعة من جملة الاسترجاع وكذلك نوع البيانات يجب أن يكون نفس نوع البيانات للقيم الراجعة .
- للتأكد من توافق أنواع البيانات نعرف المتغيرات باستخدام %TYPE .

```
v_deptno        dept.deptno%TPYPE;
v_loc           dept.loc%TPYPE;
```

- لا يشترط وجود WHERE ولكن يجب ضمان أن جملة الاسترجاع تعيد قيمة واحدة فقط.

- إذا أردت أن تستخدم عمليات الصفوف المجمعة Group Function فقم باستخدامها داخل جملة استرجاع لأن Group Function لا يمكن استخدامها في PLSQL .

مثال

```
DECLARE
  v_sum_sal   emp.sal%TYPE;
  v_deptno     NUMBER NOT NULL := 10;
BEGIN
  SELECT     SUM(sal)  -- group function
  INTO       v_sum_sal
  FROM       emp
  WHERE      deptno = v_deptno;
END;
```

معالجة البيانات باستخدام جمل DML و التحكم بالعمليات Control Transaction

تتم عملية معالجة البيانات الموجودة في قاعدة البيانات باستخدام جمل **Data Manipulation DML** وهي:

١) **جمل الإضافة INSERT** : إضافة سجل جديد إلى الجدول .

```
BEGIN
  INSERT INTO   emp(empno, ename, job, deptno)
  VALUES           (empno_sequence.NEXTVAL, 'HARDING', 'CLERK', 10);
END;
```

٢) **جمل التعديل UPDATE** : تعديل القيم الموجودة في الجدول .

- **WHERE** تستخدم لبيان الصفوف التي يجب عمل التعديل عليها
- إذا لم يتم تعديل أي صف فإن ذلك لا يؤدي إلى ظهور خطأ.

- لإسناد القيم الجديدة إلى الأعمدة نستخدم = أما لإسناد القيم للمتغيرات نستخدم **:=**

```
DECLARE
 v_sal_increase   emp.sal%TYPE := 2000;
BEGIN
 UPDATEemp
 SET             sal = sal + v_sal_increase
 WHERE         job = 'ANALYST';
END;
```

٣) **جمل الحذف DELETE**: حذف سجل من الجدول .

- إذا لم يتم حذف أي صف فإن ذلك لا يؤدي إلى ظهور خطأ.

```
DECLARE
  v_deptno   emp.deptno%TYPE := 10;
BEGIN
  DELETE       FROM   emp
  WHERE        deptno = v_deptno;
END;
```

ملحوظة : يجب استخدام أسماء متغيرات تختلف عن أسماء الأعمدة وذلك لتجنب الغموض الذي يمكن أن يحدث في جزء الشرط WHERE والذي سيؤدي بدوره إلى حدوث الأخطاء .فمثلا استخدام المتغير ordid في المثال التالي وهو نفس الاسم لعمود ordid في الجدول ord فهذا سيؤدي إلى غموض في القيم لأنه في عملية التنفيذ تتم أولا عملية تعويض القيم للأعمدة قبل المتغيرات ،ففي المثال التالي

WHERE ordid = ordid ;

لن يتم تعويض قيم المتغير ordid والتي هي 601 بل سيتم تعويض قيم العمود في قاعدة البيانات وبالتالي ستتم مقارنة كل قيمة للعمود ordid مع نفسه وهذا يؤدي إلى استرجاع جميع الصفوف في الجدول وهذا يتناقض مع أن جملة الاسترجاع يجب أم تعيد القيم من صف واحد فقط.

```
DECLARE
  orderdate         ord.orderdate%TYPE;
  shipdate          ord.shipdate%TYPE;
  ordid             ord.ordid%TYPE := 601;
BEGIN
  SELECT        orderdate, shipdate
  INTO          orderdate, shipdate
  FROM   ord
  WHERE         ordid = ordid;
END;
SQL> /
DECLARE
*
ERROR at line 1:
ORA-01422: exact fetch returns more than requested
number of rows
ORA-06512: at line 6
```

التحكم بالعمليات Control Transaction:

Transaction: هو عبارة مجموعة عمليات معالجة البيانات(إضافة ، تعديل، حذف) التي تمت خلال فترة معينة وتتضمن عملية التحكم في Transaction بتثبيت هذه العمليات أو عدم تثبيتها .من خلال الأوامر التالية :

- **COMMIT** ويعني تثبيت عمليات الإضافة والتعديل والحذف التي تمت حتى هذه اللحظة.
- **ROLLBACK** ويعني التراجع عن عمليات الإضافة والتعديل والحذف التي تمت حتى هذه اللحظة.
- **SAVEPOINT** وتعني وضع نقطة يمكن التراجع عن عمليات الإضافة والتعديل والحذف التي تمت منذ إنشاء هذه النقطة وحتى هذه اللحظة.

```
COMMIT    [WORK];
SAVEPOINT savepoint_name;
 ROLLBACK  [WORK];
ROLLBACK  [WORK]   TO  [SAVEPOINT] savepoint_name;
```

استخدام مؤشر SQL Cursor SQL

المؤشر Cursor هو عبارة عن منطقة عمل خاصة تستعمل لتنفيذ أي جملة يقوم المستخدم بتنفيذها مثل جمل الاسترجاع أو التعديل أو الحذف . وتتم عملية التحكم بهذا المؤشر من قبل النظام بعكس المؤشرات التي يقوم المستخدم بتعريفها .

وسنتناول في هذا الفصل المؤشرات(SQL CURSORS) التي يتم إنشاؤها والتعامل معها من قبل النظام .

-خصائص المؤشر :

للمؤشر عدة خصائص يمكن التعامل معه من خلالها لمعرفة وتقييم العملية التي قام المستخدم بتنفيذها وهذه الخصائص هي :

١ - SQL%ROWCOUNT تعيد عدد الصفوف التي تأثرت بآخر جملة SQL .

٢ - SQL%FOUND تعيد(TRUE)إذا تأثر صف أو أكثر بآخر جملة SQL .

٣ - SQL%NOTFOUND تعيد(TRUE)إذا لم تؤثر آخر جملة SQL بأي صف .

٤ - SQL%ISOPEN تعيد (TRUE) إذا كان المؤشر مفتوحاً.بالنسبة لـSQL CURSORS دائما تكون القيمة الراجعة FALSE لأنه يفتح ويقفل ضمنيا من قبل النظام بخلاف المؤشرات التي يقوم المستخدم بإنشائها .

مثال

```
1     SQL> VARIABLE    rows_deleted VARCHAR2(30)

2     DECLARE
3     v_ordid  NUMBER := 605;
4     BEGIN
5     DELETE FROM  item
6     WHERE       ordid = v_ordid;
7     :rows_deleted := (SQL%ROWCOUNT || ' rows deleted.');
8     END;
9     /
10    SQL >PRINT rows_deleted
```

في المثال السابق تم تعريف المتغير rows_deleted على مستوى SQL*Plus ومن ثم كتابة وحدة (Block) بحيث تقوم هذه الوحدة بحذف جميع السجلات من جدول item للطلب order رقم 605 وفي السطر ٧ تمت عملية استدعاء الخاصية **SQL%ROWCOUNT** والتي تمثل عدد الصفوف التي تم حذفها ، ثم تخزينها مع الجملة التوضيحية || ' rows deleted.' في المتغير rows_deleted. وعد ذلك طباعة المتغير من خلال SQL*Plus .

تمارين

١ - اكتب وحدة (Block) تقوم بما يلي :
- استرجاع رقم أكبر قسم من جدول DEPT.
- تخزين القيم في متغير SQL*Plus .
- طباعة المتغير من خلال الشاشة .
- حفظ الملف بالاسم p8q1.sql.

```
G_MAX_DEPTNO
------------
        40
```

٢ - قم بتعديل الملف في السؤال السابق ليقوم بإضافة صف جديد إلى جدول DEPT وقم بحفظ الملف باسم p8q2.sql . مع مراعاة ما يلي :
- بدل من طباعة اسم المتغير قم بإضافة ١٠ إلى الرقم ليكون رقم القسم الجديد .
- استخدم متغير تعويض SQL*Plus Substitution Variable لإدخال اسم القسم .
- إبقاء قيمةNULL ل location .
- قم بتنفيذ الوحدة (Block).

```
Please enter the department name: EDUCATION

PL/SQL procedure successfully completed.
```

- قم باسترجاع معلومات القسم الجديد الذي قمت بإدخاله .

```
 DEPTNO      DNAME           LOC
 ---------   --------------  -------------
   50        EDUCATION
```

٣ - اكتب وحدة (Block) ليقوم بتعديل الموقع location لقسم معين في جدول .DEPT قم بحفظ الملف باسم p8q3.sql . ثم قم بتنفيذه. مع مراعاة ما يلي :
- استخدم متغير تعويض SQL*Plus Substitution Variable لإدخال رقم القسم .
- استخدم متغير تعويض SQL*Plus Substitution Variable لإدخال الموقع location .

```
Please enter the department number: 50
Please enter the department location: HOUSTON

PL/SQL procedure successfully completed.
```

- قم باسترجاع معلومات القسم الذي قمت بتعديل موقعه .

```
DEPTNO     DNAME            LOC
----------    --------------    -------------
    50      EDUCATION    HOUSTON
```

٤ - اكتب وحدة (Block) ليقوم بحذف القسم الذي قمت بإدخاله في السؤال الثاني. قم بحفظ الملف باسم p8q4.sql . ثم قم بتنفيذه. مع مراعاة ما يلي:

- استخدم متغير تعويض SQL*Plus Substitution Variable لإدخال رقم القسم .
- قم بطباعة عدد الصفوف التي تم حذفها.

```
.Please enter the department number: 50
PL/SQL procedure successfully completed.

G_RESULT
--------------------------------------------------------------------------------
1 row(s) deleted.
```

- ما هي النتيجة عند إدخال رقم قسم غير موجود ؟

```
Please enter the department number: 99
PL/SQL procedure successfully completed.

G_RESULT
--------------------------------------------------------------------------------
0 row(s) deleted.
```

- تأكد من أن القسم قد تم حذفه فعليا من الجدول .

تصميم قواعد البيانات

جمل التحكم

جمل التحكم

٩

الجدارة:

القدرة على كتابة التراكيب التي تتحكم بسير عملية التنفيذ داخل الوحدة.

الأهداف:

- أن يتعرف المتدرب على أنواع جمل التحكم بسير التنفيذ.
- أن يستخدم المتدرب جمل الشرط بمختلف أنواعها.
- أن يستخدم المتدرب حلقات الدوران بمختلف أنواعها.
- أن يستخدم المتدرب جداول الصدق.
- أن يستخدم المتدرب حلقات الدوران المتداخلة.

مستوى الأداء المطلوب:

أن يتقن المتدرب عملية التحكم بسير البرنامج بنسبة ١٠٠٪.

الوقت المتوقع للتدريب:

٣ ساعات

الوسائل المساعدة:

- معمل حاسب آلي
- قلم + دفتر

متطلبات الجدارة:

أن يكون المتدرب قد أتقن الجدارة في الوحدات السابقة .

مقدمة :

تتم عملية التنفيذ للوحدة (Block) وذلك بتنفيذ الجمل بنفس الترتيب الذي كتبت فيه هذه الجمل ولكن قد تبرز الحاجة في معظم المسائل لتغيير سير التنفيذ بناء على معطيات معينة فمثلا إذا أردنا أن نقوم بزيادة الراتب بنسبة ١٠٪ لكل موظف راتبه أقل من ٣٠٠٠ وزيادة الراتب بنسبة ٨٪ لكل موظف راتبه اكبر من أو يساوي ٣٠٠٠. فلذلك لابد أن يتغير سير التنفيذ بناء على قيم الراتب. وكذلك قد تتطلب بعض المسائل تكرار عدد من الجمل وهنا أيضا نضطر إلى تغيير سير عملية التنفيذ.

جملة الشرط البسيطة **IF Statement** .

تتكون جملة IF البسيطة من جملة الشرط يليها مجموعة الجمل الواجب تنفيذها عند تحقق هذا الشرط.

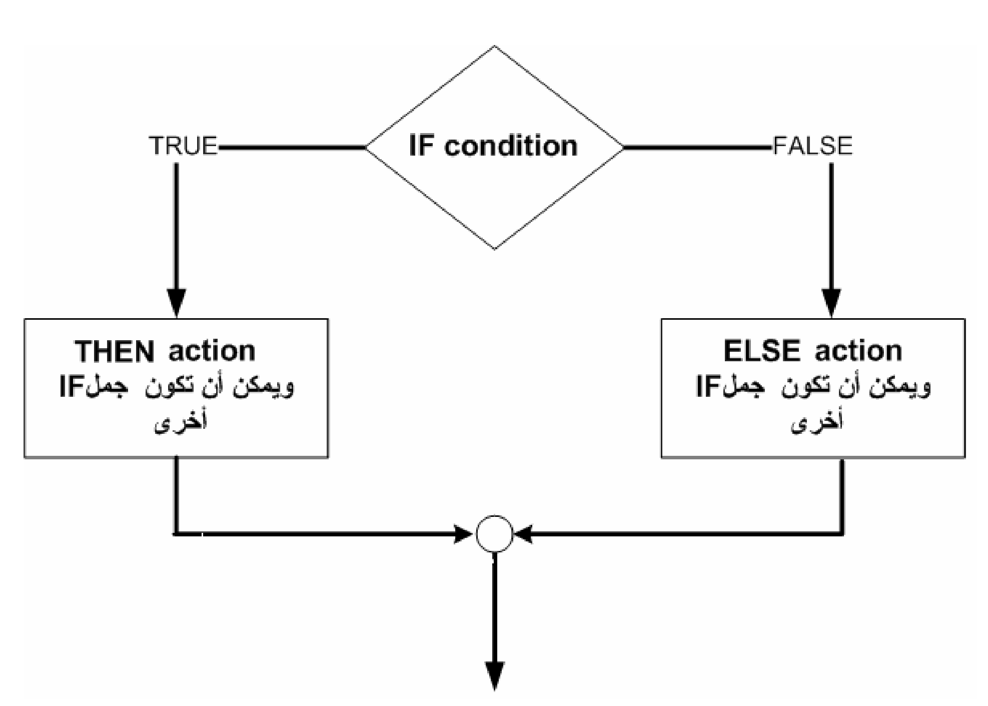
```
IF condition THEN
  statements;
 END IF;
```

في المثال التالي نقوم بإسناد ٣٥٠٠ للمتغير v_sal اذا كانت قيمة المتغيرv_ename تساوي Ali أي إن نتيجة جملة الشرط هي True و سيتم تنفيذ الجملة(الجمل) التالية في هذه الحالة v_sal := . 3500وإذا لم تكن قيم المتغير v_ename تساوي Ali ستكون نتيجة جملة الشرط هي False بالتالي لن يتم تنفيذ الجملة ;v_sal := 3500 وسينتقل التنفيذ إلى الجملة التي تليEND IF :

```
….
IF v_ename = 'Ali'  THEN
  v_sal := 3500;
END IF;
…….
```

جمل الشرط IF THEN ELSE :

تتكون جملة IF THEN ELSE من مجموعتين من الجمل الأولى وهي الجمل الواجب تنفيذها في حالة تحقق الشرط والأخرى مجموعة الجمل الواجب تنفيذها في حالة عدم تحقق الشرط.




```
IF CONDITION1 THEN
 Statement1;
ELSE
Statement2.
END IF;
```

```
IF v_deptno = 10 THEN
      UPDATE emp
      SET sal = sal * 1.10
      WHERE  deptno = v_deptno;
ELSE
      UPDATE emp
      SET sal = sal * 1.08
      WHERE  deptno = v_deptno;
END IF;
```

في المثال يتم السؤال عن رقم القسم فإذا كان رقم القسم يساوي ١٠ فيتم زيادة رواتب الموظفين بذلك القسم بنسبة ١٠٪، وإذا كان رقم القسم غير ذلك (لم يتحقق الشرط) سيتم زيادة رواتب الموظفين في ذلك القسم بنسبة ٨٪ .

```
IF CONDITION1 THEN
 Statement1;
 ELSE
```

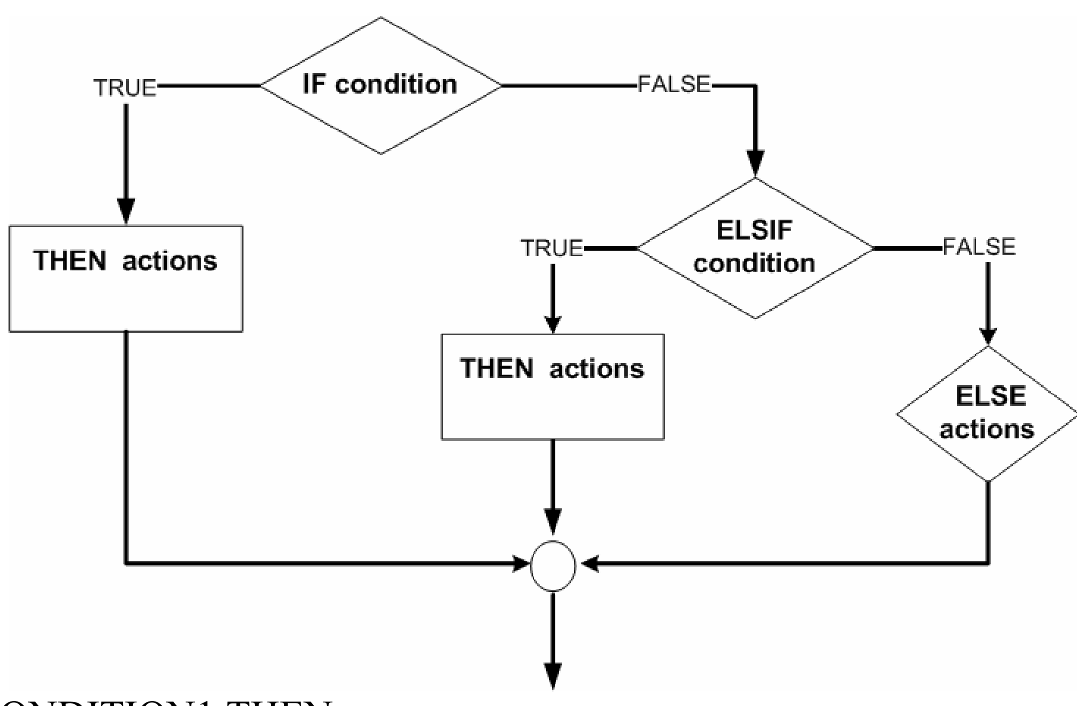
```
   IF CONDITION2 THEN
   Statement2;
    END IF;
END IF;

IF v_deptno = 10 THEN
      UPDATE emp
      SET sal = sal * 1.10
      WHERE  deptno = v_deptno;
ELSE
    IF v_job = 'SALESMAN' THEN
      UPDATE emp
      SET sal = sal * 1.11
      WHERE  job =  v_job;
END IF;
```

في المثال يتم السؤال عن رقم القسم فإذا كان رقم القسم يساوي ١٠ فيتم زيادة رواتب الموظفين بذلك القسم بنسبة ١٠٪ ، وإذا كان رقم القسم غير ذلك (لم يتحقق الشرط) سيتم السؤال عن الوظيفة إذا كانت SALESMAN فسيتم زيادة رواتب الموظفين الذين يعملون في تلك الوظيفة بنسبة ٨٪ .

جملة الشرط IF THEN ELSIF :

تتكون جملة IF THEN ELSIF من مجموعتين من الجمل الأولى وهي الجمل الواجب تنفيذها في حالة تحقق الشرط والأخرى جملة IF جديدة وتكون على الشكل التالي (ELSIF) وهذه بدورها يمكن أن تتكون أيضا من جزأين الأول مجموعة الجمل الواجب تنفيذها عند تحقق الشرط والأخرى يمكن أن تكون جملة IF جديدة حتى لا تكون هناك أي جملة IF جديدة ويمكن لآخر جملة IF أن تحتوي على جزء عدم تحقق الشرط ESLE .




```
IF CONDITION1 THEN
 Statement1;
ELSIF CONDITION2 THEN
   Statement2;
ELSIF CONDITION3 THEN
   Statement3;
.
.
.
ELSE
   StatementN;
END IF;
IF v_grade >100 OR  v_grade < 0 THEN
     DBMS_OUTPUT.PUT_LINE('Invalid Grade ');
ELSEF v_grade >= 90 THEN
     DBMS_OUTPUT.PUT_LINE('A');
ELSIF v_grade >= 80 THEN
     DBMS_OUTPUT.PUT_LINE('B');

ELSIF v_grade >= 70 THEN
     DBMS_OUTPUT.PUT_LINE('C');
ELSIF v_grade >= 60 THEN
```

```
    DBMS_OUTPUT.PUT_LINE('D');
ELSE
    DBMS_OUTPUT.PUT_LINE('F');
END IF;
```

في المثال السابق يتم السؤال عن قيمة المتغير v_grade فإذا كانت قيمة المتغير أقل من ٠ أو أكبر من ١٠٠ فإنه يتم طباعة رسالة تخبر المستخدم بأن القيمة غير مقبولة وأنها يجب أن تكون بين ٠ و١٠٠ .أما إذا كانت قيمة المتغير بين ٠ و ١٠٠ فإنه يتم طباعة الرمز الذي يقابل تلك القيم حسب الجدول التالي :

الدرجة	التقدير
٩٠ ـ ١٠٠	A
٨٠ ـ ٨٩	B
٧٠ ـ ٧٩	C
٦٠ ـ ٦٩	D
٠ -٥٩	F

استخدام جداول الصدق:

كما لاحظنا فإن عملية تنفيذ جملة IF تعتمد على الشرط الموجود في تلك الجملة ولكي نتمكن من كتابة جمل IF لابد أن نتعرف على كيفية بناء الشرط حتى نستطيع حل المسألة بشكل صحيح ولتنفيذ ذلك يجب مراعاة ما يلي :

- السؤال عن القيم لمعرفة ما إذا كانت هذه القيمة تساوي NULL نستخدم التعبير IS NULL

```
IF    v_name    IS NULL THEN
…
….
END IF
```

- أي تعبير حسابي يحتوي على قيمة NULL فإن نتيجة التعبير تكون NULL .

```
v_sal := 3000;
v_comm ; v_ annual_salary := 12 * v_sal +
```

لو فرضنا أن قيمة المتغير v_comm في المثال السابق هي NULL فإن قيمة المتغير v_ annual_salary تكون NULL .

- إذا تم ربط قيم رمزية مع NULL فإن NULL تعامل في هذه الحالة على أنها فارغة .

```
  …
  v_job := NULL;
```

```
v_name:='AHMED';
v_info:= v_name || '  is  ' || v_job ;
....
```

في المثال السابق تكون قيمة المتغير v_info هي AHMED is فقط أي إن v_job تمت معاملتها على أنها فارغة .

- نقوم بعملية ربط أكثر من شرط باستخدام العمليات المنطقية AND ,OR , NOT

```
IF v_sal > 3000 AND  v_job = 'SALESMAN' OR  v_deptno = 10  THEN
 .....
END IF ;
```

ولمعرفة نتيجة تنفيذ شرط مركب من أكثر من جزء نستخدم الجداول التالية :

AND	**TRUE**	**FLASE**	**NULL**
TRUE	**TRUE**	**FLASE**	**NULL**
FLASE	**FLASE**	**FLASE**	**FLASE**
NULL	**NULL**	**FLASE**	**NULL**

OR	**TRUE**	**FLASE**	**NULL**
TRUE	**TRUE**	**TRUE**	**TRUE**
FLASE	**TRUE**	**FLASE**	**NULL**
NULL	**TRUE**	**NULL**	**NULL**

NOT	
TRUE	**FLASE**
FLASE	**TRUE**
NULL	**TRUE**

مثال

v_flag := v_reorder_flag AND v_reorder_flag ;

ما هي قيمة v_flag في كل من الحالات التالية :

v_flag	v_reorder_flag	v_reorder_flag
TRUE	TRUE	TRUE
FALSE	FALSE	TRUE
NULL	TRUE	NULL
FALSE	FALSE	NULL

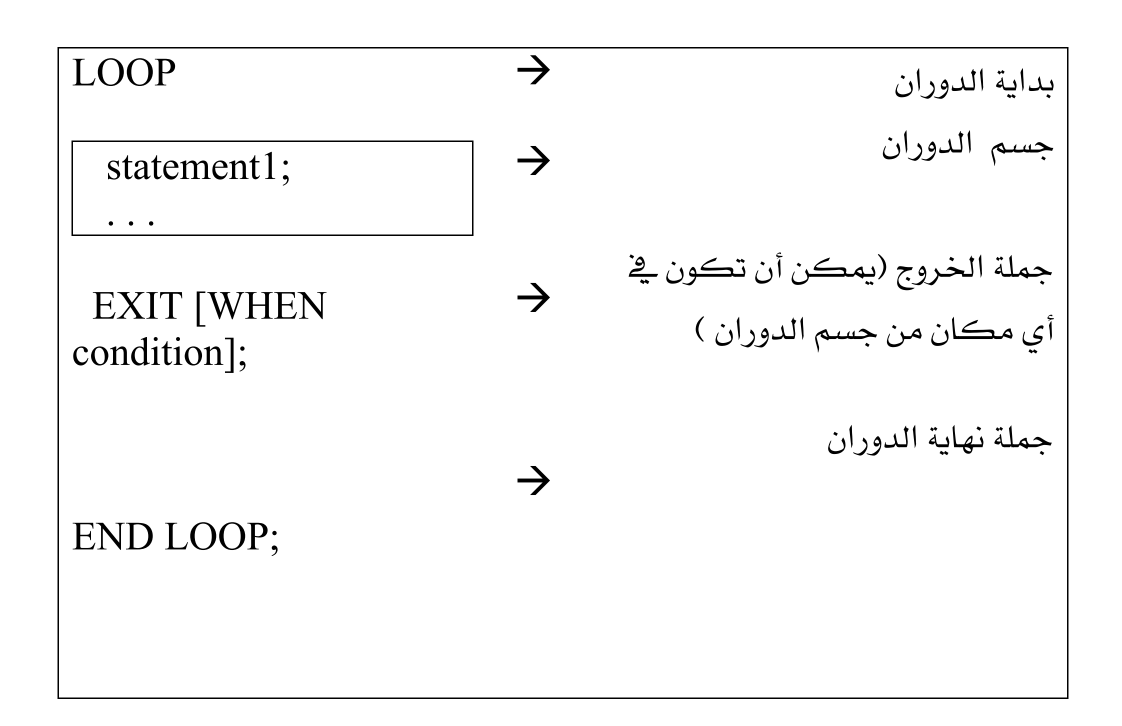
حلقات الدوران Loops.

توفر لغة PL /SQL إمكانية استخدام الدوران وهي عبارة عن تكرار جملة أو مجموعة من الجمل،وتبرز أهمية هذه الخاصية لأن هناك الكثير من المسائل التي تحتاج إلى تكرار جملة أو عدد من الجمل لحل هذه المسألة . وتوفر لغة PL /SQL عدة أشكال لصيغة ألدوران :

- حلقة الدوران البسيطة Basic Loop.
- حلقة الدوران FOR .
- حلقة الدوران WHILE .
- حلقات الدوران المتداخلة Nested LOOPS.

حلقة الدوران البسيطة Basic Loop :

وتتكون هذه الحلقة من جملة بداية الدوران Loop وتنتهي بجملة نهاية الدوران End Loop ،وفي هذه الحالة ستتم عملية تنفيذ الجمل الواقعة مابين بداية الدوران ونهايته(جسم الدوران) عدد لانهائي من المرات (أي لن يتوقف الدوران أبدا) ولحل هذه المشكلة فلابد من أن يحتوي جسم الدوران على جملة Exit وهي عبارة عن الشرط الذي يجب تحققه لإنهاء الدوران وعند الخروج من الدوران سينتقل التنفيذ إلى الجملة التي تلي جملة End Loop .



مثال

```
DECLARE

item.ordid%TYPE := 601;      v_ordid
NUMBER(2) := 1;          v_counter

BEGIN

  LOOP

    INSERT INTO item(ordid, itemid)
    VALUES(v_ordid, v_counter);
    v_counter := v_counter + 1;
    EXIT WHEN v_counter > 10;

END LOOP;

END;
```

في المثال السابق ستتم عملية إضافة ١٠ items إلى جدول item وذلك من خلال جملة الدوران أي ستتم عملية الدوران حتى تصبح قيمة v_counter أكبر من ١٠ .

مثال : حدد عدد مرات كل من جمل الدوران التالية :

أ -

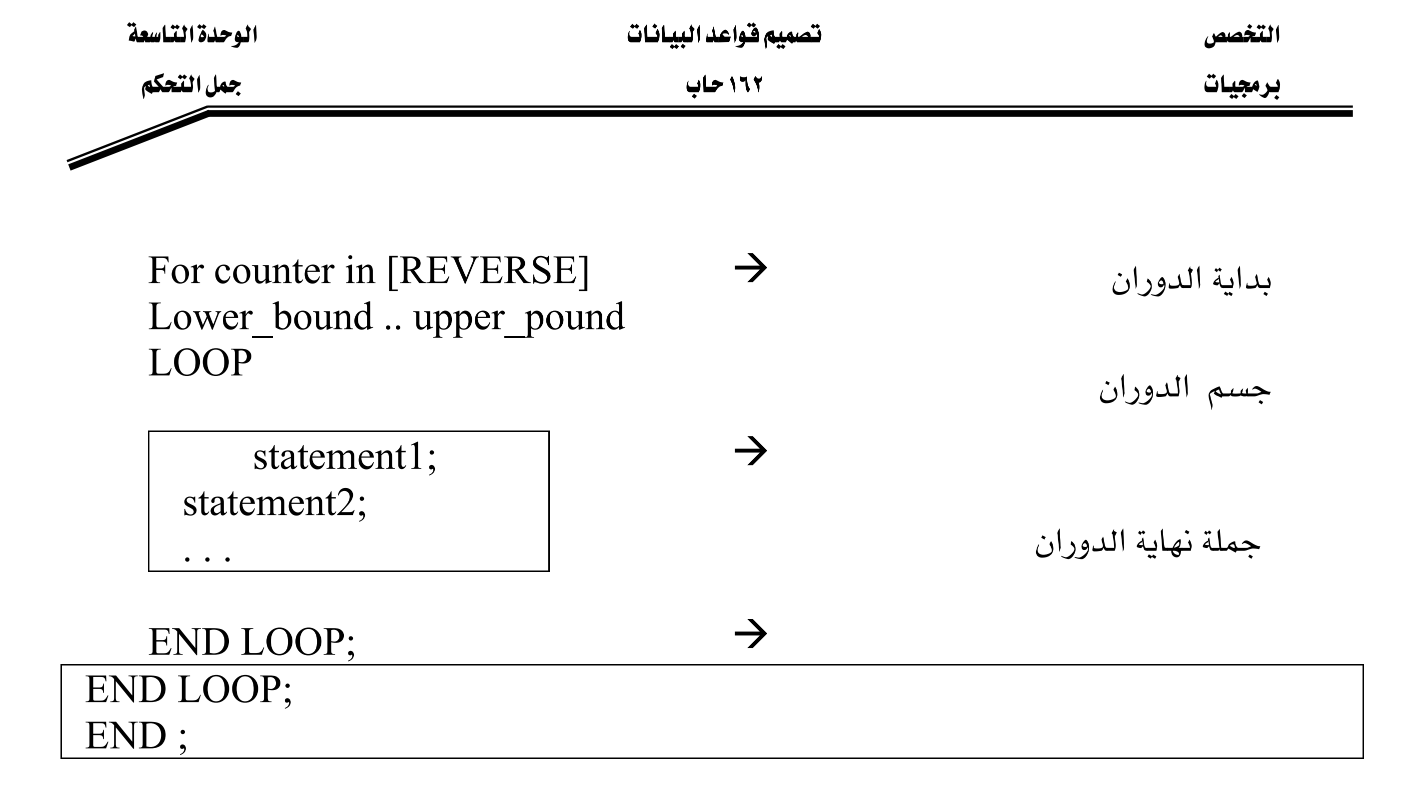
```
DECLARE
 v_counter NUMBER :=0;
 BEGIN
 LOOP
 DBMS_OUTPUT.PUT_LINE('v_counter = '||v_counter);
 EXIT WHEN v_counter > 5;
 v_counter:=v_counter+1;
 END LOOP;
 END ;
```

ب -

```
DECLARE
 v_counter NUMBER :=0;
 BEGIN
 LOOP
 DBMS_OUTPUT.PUT_LINE('v_counter = '||v_counter);
 EXIT WHEN v_counter > 5;
 v_counter:=v_counter-1;
 END LOOP;
 END ;
```

ج -

```
DECLARE
 v_counter NUMBER :=10;
 BEGIN
 LOOP
 DBMS_OUTPUT.PUT_LINE('v_counter = '||v_counter);
 EXIT WHEN v_counter > 5;
 v_counter:=v_counter+1;
```




```
END LOOP;
END ;
```

أ -	٧ مرات
ب -	عدد لانهائي من المرات
ج -	١ مرة واحدة

سؤال :ما هي عدد مرات تكرار كل من الوحدات () السابقة عل فرض أن جملة شرط الخروج كانت بعد جملة Loop مباشرة ؟

أ -	
ب -	
ج -	

حلقة الدوران FOR :

تستخدم حلقة الدوران FOR في الحالات التي يكون فيها عدد مرات التكرار المطلوب تنفيذها معروف قبل عملية التنفيذ ١٠ مرات ، ١٥ مرة ، ... إلخ .

Counter -عبارة عن متغير يعرّف ضمنيا ويعطى قيمة ابتدائية Lower_bound وتزاد قيمته بمقدار ١ في بعد كل دورة حتى يصل إلى الحد الأعلى لمرات الدوران upper_pound (وإذا استخدمنا reverse فتنقص قيمته بمقدار واحد).

- يجب عدم تعريف هذا المتغير لأنه يعرف ضمنيا .
- يمكن استخدامه فقط داخل جسم الدوران فقط لأنه غير معرّف خارج جسم الدوران.
- لا يسمح بتغيير قيمة هذا المتغير .

REVERSE - تستخدم لإنقاص قيمة المتغير إذا أردنا أن نبدأ الدوران بطريقة عكسية من الأعلى إلى الأدنى

Lower_cound - الحد الأدنى لقيمة بداية الدوران يجب أن تكون عدداً صحيحاً .
- يمكن أن تكون قيمة ثابتة أو متغيرة أو تعبيراً حسابياً (ويجب أن تؤدي إلى عدد صحيح).

upper_cound -الحد الأعلى لقيمة نهاية الدوران يجب أن تكون عدداً صحيحاً
- يمكن أن تكون قيمة ثابتة أو متغيرة أو تعبيراً حسابياً (ويجب أن تؤدي إلى عدد صحيح).

```
  BEGIN
   FOR   i IN   1..5   LOOP
    DBMS_OUTPUT.PUT_LINE('i= '||i);
   END LOOP;
  END;
SQL> /
i= 1
i= 2
i= 3
i= 4
i= 5
```

ويمكن كتابة ال Block السابق على الشكل التالي :

```
 DECLARE
 v_lower      number:=1;
 v_upper     number:=5;
BEGIN
   FOR   i IN   v_lower.. v_upper   LOOP
    DBMS_OUTPUT.PUT_LINE('i= '||i);
   END LOOP;
  END;
SQL> /
i= 1
i= 2
i= 3
i= 4
i= 5
```

حلقة الدوران WHILE :

تستخدم حلقة الدوران WHILE في الحالات التي لا يكون فيها عدد مرات تكرار الدوران معروف، وتستمر عملية الدوران مادام شرط الدوران متحقق .

WHILE (condition)	→	بداية الدوران
statement1; . . .	→	جسم الدوران
END LOOP;	→	جملة نهاية الدوران

أ -

```
DECLARE
 v_counter NUMBER :=0;
 BEGIN
 WHILE (v_counter <= 5) LOOP
 DBMS_OUTPUT.PUT_LINE('v_counter = '||v_counter);
  v_counter:=v_counter+1;
 END LOOP;
 END ;
```

وتشبه حلقة الدوران while حلقة الدوران البسيطة في الحالة التي تكون جملة الخروج في بداية جسم الدوران ولكن بعكس الشرط.فلو قمنا بتنفيذ كلا الوحدتين فستكون نتيجة تنفيذهما واحدة.

```
DECLARE
 v_counter NUMBER :=0;
 BEGIN
 LOOP
 DBMS_OUTPUT.PUT_LINE('v_counter = '||v_counter);
 EXIT WHEN v_counter > 5;
 v_counter:=v_counter+1;
 END LOOP;
 END ;
```

واجب صفي : قم بإعادة كتابة الأمثلة في جملة الدوران البسيطة على شكل حلقات دوران WHILE

مثال : يقوم بطلب من المستخدم بإدخال رقم الطلب وعدد مفردات هذا الطلب ثم القيام بعملية إدخالها في جدول item

```
ACCEPT p_new_order PROMPT 'Enter the order number: '
ACCEPT p_items -
  PROMPT 'Enter the number of items in this order: '
DECLARE
NUMBER(2) := 1;              v_count
BEGIN
  WHILE v_count <= &p_items LOOP
    INSERT INTO item (ordid, itemid)
    VALUES (&p_new_order, v_count);
    v_count := v_count + 1;
  END LOOP;
  COMMIT;
END;
```

حلقات الدوران المتداخلة Nested Loops :

نستطيع كتابة عدة مستويات من الدوران داخل بعضها ويمكن كتابة عدة مستويات لمختلف حلقات الدوران (البسيطة Basic Loops ، حلقة FOR و حلقة WHILE) . ولتمييز هذه المستويات عن بعضها باستخدام عنوان (Label) لكل مستوى من هذه المستويات . و للخروج من الدوران الخارجي لابد من ذكر عنوان(Label) الدوران صراحة لأن جملة الخروج(Exit) دون عنوان تؤدي للخروج من الدوران الداخلي ومن ثم ينتقل التنفيذ إلى بداية الدوران الخارجي لبدء دورة جديدة .

```
BEGIN
<<Outer_loop>>  عنوان الدوران الخارجي
  LOOP
    v_counter := v_counter+1;
  EXIT WHEN v_counter>10;
<<Inner_loop>>    عنوان الدوران الداخلي
    LOOP
      ...
      EXIT Outer_loop WHEN total_done = 'YES';
-- Leave both loops        الخروج من كلا الدورانين
      EXIT WHEN inner_done = 'YES';
-- Leave inner loop only        الخروج من الدوران الداخلي فقط
      ...
    END LOOP Inner_loop;
    ...
  END LOOP Outer_loop;
END;
```

تمارين

١ - قم بإنشاء الجدول التالي لاستخدامه في الحل:

```
CREATE TABLE messages
VARCHAR2(60))      (results
```

-قم بإدخال القيم من ١ ... ١٠ في الجدول مستثنيا القيم ٦ ، ٨ .(استخدم جملة التكرار FOR).

- قم بتثبيت عملية التخزين في قاعدة البيانات داخل الوحدة (Block) .

- قم باسترجاع جميع محتويات الجدول messages .

```
  RESULTS
----------
       1
       2
       3
       4
       5
       7
       9
      10
```

٢ - قم بإضافة السجل التالي لجدول emp ،لاحظ أن قيمة الراتب salary هي NULL:

```
insert into emp
values (8000, 'DOE', 'CLERK', 7698, SYSDATE, NULL,
NULL, 10);
```

- قم بكتابة وحدة (PL/SQL Block) لإضافة العمولة لكل موظف بناء على راتبه بناء على المعطيات التالية :

أ - قم بقراءة رقم الموظف باستخدام substitution variable .

ب - قم بحساب العمولة(commission) وتثبتها في قاعدة البيانات حسب الجدول التالي :

الراتب	نسبة العمولة
أقل من ١٠٠٠	١٠٪
١٠٠٠ -١٥٠٠	١٥٪
اكبر من ١٥٠٠	٢٠٪
NULL	صفر

ج - قم بعملية استرجاع لجدول emp للتأكد من عملية التعديل .

٣ - قم بتعديل الملف p6q4.sql ليقوم بإضافة ('The number is odd') إلى جدول message إذا كان الرقم فردياً أو إضافة ('The number is even') إلى الجدول إذا كان الرقم زوجيا .

١. قم بإضافة عمود STARS جديد إلى جدول emp .

٢. قم بإنشاء وحدة (PL/SQL Block) لإضافة * في العمود STARS لكل ١٠٠ من الراتب (إذا كان راتب الموظف ١٠٠٠ فيجب أن نضع في STARS **********) .قم بتخزين الوحدة في ملف p9q3.sql .

- قم بقراءة رقم الموظف باستخدام substitution variable .

- قم بالتأكد من أن التعديلات قد تمت بشكل صحيح .



تصميم قواعد البيانات

معالجة الاستثناءات

معالجة الاستثناءات

١٠

الجدارة:

القدرة على تعريف واستخدام الاستثناءات المختلفة.

الأهداف:

- أن يتعرف المتدرب الستثناءت.
- أن يتعرف المتدرب أنواع الاستثناءات و كيفية معالجتها.
- أن يقوم المتدرب بتعريف ومعالجة الاستثناءات.

مستوى الأداء المطلوب:

أن يتقن المتدرب تعريف واستخدام الاستثناءات المختلفة بنسبة ١٠٠٪.

الوقت المتوقع للتدريب:

3 ساعات

الوسائل المساعدة:

- معمل حاسب آلي.
- قلم + دفتر.

متطلبات الجدارة:

أن يكون المتدرب قد أتقن الجدارة المطلوبة في الوحدات السابقة.

مقدمة :

الاستثناءات هي عبارة عن خطأ يظهر خلال عملية تنفيذ الوحدة Block ويؤدي إلى وقف تنفيذ الجز التنفيذي من الوحدة .وفي جميع الحالات التي يظهر فيها الاستثناء يتم وقف تنفيذ الوحدة ولكن إذا قمنا بعملية معالجة للاستثناء فيمكن عمل بعض الإجراءات قبل عمل الإيقاف، وتتم عملية إظهار الاستثناء عند حدوث خطأ (Error Oracle).أو يتم إظهار الاستثناء صراحة من قبل المستخدم.أما معالجة الاستثناءات فيمكن أن نقوم بها داخل الوحدة ،أو نتركها بدون معالجة وترك أمر المعالجة إلى البيئة التي قامت باستدعاء هذه الوحدة (Block) .

أنواع استثناءات :

هي عبارة أخطاء تحدث في الوحدة خلال عملية التنفيذ وتؤدي إلى توقف عملية التنفيذ.

وتقسم الاستثناءات إلى ثلاثة أقسام:

١ - **الأخطاء المعرفة مسبقا Predefined Oracle Server errors :**

وهي عبارة عن ٢٠ خطأ والتي كثيرا ما تكرر في البرامج .وبالنسبة لهذه الأخطاء يجب عدم إظهارها (لا تقم بعمل RAISE لها) لأنها تُظهر ضمنيا من قِبل خادم Oracle (Oracle Server) .

٢ - **الأخطاء غير المعرفة مسبقا Non-Predefined Oracle server errors :**

وهي عبارة عن أي خطأ من أخطاء oracle غير تلك المعرفة مسبقا .وهذه يجب أن تعرف في جزء التعريف في الوحدة (Block), وأما عملية إظهارها فتتم بشكل ضمني من قِبل خادم Oracle (Oracle Server) .

٣ - **استثناءات المستخدم User Defined Exceptions:**

وهي عبارة عن أي حدث يعتبره المستخدم على أنه خطأ ويجب وقف تنفيذ الوحدة(Block) نتيجة حدوث هذا الخطأ . وهذه الاستثناءات تعرّف وتُظهر صراحة من قِبل المستخدم .

ولمعالجة أي من هذه الاستثناءات لابد من الإمساك به وتتم عملية الإمساك بأي استثناء باستخدام في WHENمتبوعة باسم الاستثناء المراد معالجته في جزء الاستثناءات في الوحدة (Block).ثم بعد ذلك كتابة الإجراءات التي يجب القيام بها قبل عملية الانتهاء نتيجة لحدوث هذا الخطأ. ويمكن أن نقوم بكتابة WHEN OTHERS THEN وتعني إذا حدث أي خطأ غير الأخطاء السابقة فقم بما يلي. يجب أن تكون WHEN OTHERS بعد السؤال عن جميع الاستثناءات التي يمكن أن

تظهر.وكذلك يجب مراعاة ترتيب وضع جمل WHEN للإمساك بالاستثناءات حسب إمكانية حدوثها لأن عملية المعالجة لا تتم إلا لاستثناء واحد من هذه الاستثناءات قبل عملية إنهاء عملية التنفيذ .

```
EXCEPTION
  WHEN exception1 [OR exception2 . . .] THEN
    statement1;
    statement2;
    . . .
  [WHEN exception3 [OR exception4 . . .] THEN
    statement1;
    statement2;
    . . .]
  [WHEN OTHERS THEN
    statement1;
    statement2;
    . . .]
```

الاستثناءات للأخطاء المعرفة مسبقا.

وهي كما ذكرنا سابقا عبارة عن الاستثناءات التي تحدث نتيجة حدوث أحد الأخطاء الشائعة التي يمكن أن تظهر خلال عملية التنفيذ.وتتم عملية الإمساك بهذه الأخطاء في جزء الاستثناءات في الوحدة (Block).

ولكل خطأ من هذه الأخطاء اسم وتتم عملية إظهار هذه الاستثناءات ضمنيا من قبل خادم oracle (Oracle Server). ولمعالجة هذه الاستثناءات. عند حدوثها نقوم بمحاولة الإمساك بها في جزء الاستثناءات في الوحدة (Block) ومن ثم كتابة الإجراء الذي يجب عمله عند ظهور استثناء معين.والجدول التالي يبين هذه الاستثناءات مع وصف مبسط للخطأ المسبب لظهورها.

الاستثناء	الوصف
ACCESS_INTO_NULL	محاولة وضع قيمة لصفة كائن لم يعمل له initialization
COLLECTION_IS_NULL	محاولة تطبيق collection method على nested tables أو varray لم يعمل لها initialization .
CURSOR_ALREADY_OPEN	المؤشر مفتوح
DUP_VAL_ON_INDEX	محاولة وضع قيم متماثلة
INVALID_CURSOR	عملية مؤشر غير صحيحة
LOGIN_DENIED	الدخول ممنوع خطأ في اسم المستخدم أو كلمة المرور
NO_DATA_FOUND	جملة استرجاع Select لم تعد أي نتيجة .
NOT_LOGGED_ON	محاولة إجراء عملية على قاعدة البيانات بدون دخول
PROGRAM_ERROR	حدوث خطأ داخلي
ROWTYPE_MISMATCH	عدم توافق في جملة الإسناد لمؤشر المضيف مع مؤشر PL/SQL
STORAGE_ERROR	وجود خطأ في الذاكرة أو أن الذاكرة ممتلئة
SUBSCRIPT_BEYOND_COUNT	محاولة الوصول إلى عنصر في collection method علىnested tables أو varray خارج المكان المسموح به .
TIMEOUR_RESOURCE	انتهاء وقت الانتظار المحدد لانتظار أحد المصادر
TOO_MANY_ROWS	جملة الاسترجاع Selectأعادت أكثر من صف
VALUE_ERROR	خطأ في عملية حسابية أو عملية تحويل أو خطأ في الحجم
ZERO_DEVIDE	القسمة على صفر

```
DECLARE
V_ename      emp.ename%Type;
V_empno      emp.empno%Type :=  &p_eno;
Begin
SELECT   ename   INTO
v_ename
From        emp
WHERE   empno   =   v_empno;
DBMS_OUTPUT.PUT_LINE ('Employee Name is ' || v_ename );

EXCEPTION

WHEN NO_DATA_FOUND   THEN
DBMS_OUTPUT.PUT_LINE (' Invalid Employee Number ' || v_empno);
END;
```

وعند تنفيذ هذه الوحدة بإدخال القيم التالية ٧٧٨٨ ، ٧٧٧٧ سنلاحظ كيفية التعامل مع الاستثناء كون جملة الاسترجاع للموظف رقم ٧٧٧٧ لن تعيد أي صفوف .

```
Enter value for p_eno: 7788

Employee Name is SCOTT

PL/SQL procedure successfully completed.

SQL> /
Enter value for p_eno: 7777

Invalid Employee Number 7777

PL/SQL procedure successfully completed.
```

مثال

```
DECLARE
V_ename       emp.ename%Type;
V_job      emp.job%Type := upper( '&p_ejob');
Begin
 SELECT    ename   INTO
 v_ename
 From        emp
 WHERE   job=   v_job;
DBMS_OUTPUT.PUT_LINE ('Employee Name is ' || v_ename );
EXCEPTION

WHEN NO_DATA_FOUND   THEN
DBMS_OUTPUT.PUT_LINE ('There is no Job Employee  has this   '|| V_job
);

WHEN TOO_MANY_ROWS  THEN
DBMS_OUTPUT.PUT_LINE ('The Job  '|| V_job  ||  '   has more than one
Employee  '   );

END;
```

وعند تنفيذ هذه الوحدة بإدخال القيم التالية PRESIDENT ،MANAGER ،DRIVER سنلاحظ كيفية التعامل مع الاستثناءات كون جملة الاسترجاع للوظيفةMANAGER ستعيد أكثر من صف وكذلك بالنسبة للوظيفة DRIVER لن تعيد أي صفوف .

```
  Enter value for p_ejob: PRESIDENT

old   3:  V_job      emp.job%Type := upper( '&p_ejob');
new   3:  V_job      emp.job%Type := upper( 'PRESIDENT');

Employee Name is KING

PL/SQL procedure successfully completed.

SQL> /
```

```
Enter value for p_ejob: MANAGER

old   3:  V_job      emp.job%Type := upper( '&p_ejob');
new   3:  V_job      emp.job%Type := upper( 'MANAGER');

The Job  MANAGER   has more than one  Employee

PL/SQL procedure successfully completed.

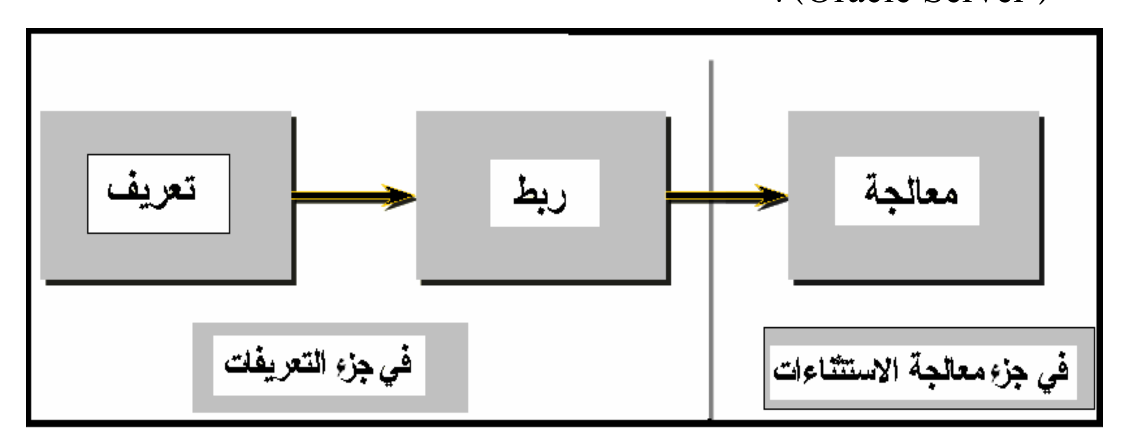
SQL> /
Enter value for p_ejob: DRIVER


There is no Job Employee  has this   DRIVER

PL/SQL procedure successfully completed.
```

الاستثناءات للأخطاء غير المعرفة مسبقا:

وهي عبارة عن أي خطأ من أخطاء oracle غير تلك المعرفة مسبقا.وهذه يجب أن تعرف في جزء التعريف في الوحدة (Block), وأما عملية إظهارها فتتم بشكل ضمني من قِبل خادم oracle (Oracle Server).



وللتعامل مع هذا النوع من الاستثناءات لابد من أن نقوم بما يلي :

١. تعريف الاستثناء في جزء التعريفات في الوحدة (Block):

```
DECLARE
.....
Exception_name    EXCEPTION;
```

٢. ربط الاستثناء مع الخطأ باستخدام **PRAGMA_EXCEPTION_INIT** وتتم هذه العملية بعد تعريف الاستثناء لتكون مؤشراً للمترجم للتعامل مع الاستثناء في أي مكان من الوحدة على أنه الخطأ الذي تم ربطه معه.

PRAGMA_EXCEPTION_INIT(*Exception_name* ,*error_number*);

٣. معالجة الاستثناء في جزء الاستثناءات في الوحدة (Block):

- محاولة الإمساك بالاستثناء وذلك بكتابة اسم الاستثناء بعد كلمة WHEN ومن ثم كتابة الجمل المناسبة للمعالجة.

-لا تقم بعملية إظهار الاستثناء بشكل صريح لأنه يظهر ضمنيا عند حدوث الخطأ الذي تم ربطه معه.

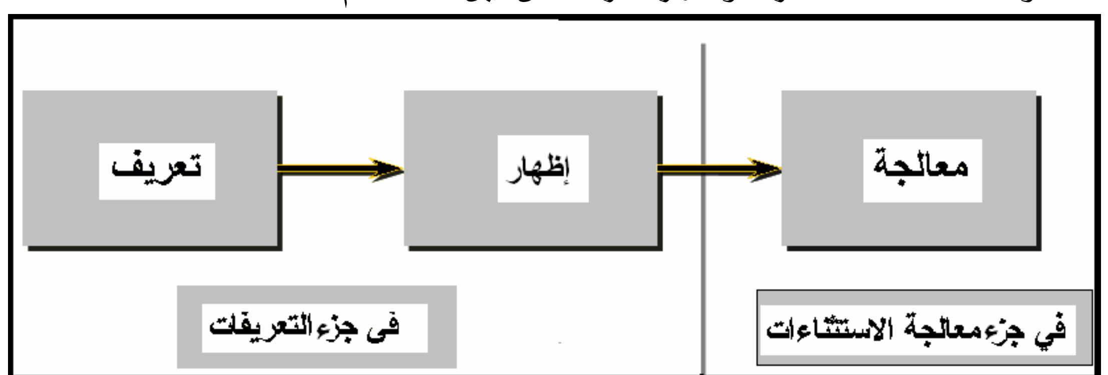
مثال :

DECLARE		
EXCEPTION; e_emps_remaining	→	تعريف الاستثناء
PRAGMA EXCEPTION_INIT (e_emps_remaining , -2292);	→	ربط الاستثناء مع الخطأ رقم ٢٢٩٢-
dept.deptno%TYPE := &p_deptno; v_deptno BEGIN DELETE FROM dept deptno = v_deptno; WHERE COMMIT; EXCEPTION		
WHEN e_emps_remaining THEN DBMS_OUTPUT.PUT_LINE ('Cannot remove dept ' || TO_CHAR(v_deptno) || '. Employees exist. ');	→	معالجة الاستثناء
END;		

في المثال السابق تم تعريف الاستثناء e_emps_remaining وربطة مع الخطأ رقم ٢٢٩٢ - وهذا الخطأ الذي يظهر عند حذف صف من جدول مع وجود قيمة في هذا الصف يشار إليها كمفتاح أجنبي في جدول آخر ،وبالتالي لا يمكن أن تتم عملية الحذف .

الاستثناءات المعرفة من قِبل المستخدم :

كما مر معنا سابقا فإن استثناءات المستخدم هي الاستثناءات التي يقوم المستخدم بتعريفها وهي عبارة عن أي حدث يعتبره المستخدم على أنه خطأ ويجب وقف تنفيذ الوحدة(Block) نتيجة حدوث هذا الخطأ . وهذه الاستثناءات تعرّف وتُظهر صراحة من قِبل المستخدم .



وللتعامل مع هذا النوع من الاستثناءات لابد من أن نقوم بما يلي :

١ - تعريف الاستثناء في جز التعريفات في الوحدة (Block):

```
DECLARE
…..
Exception_name    EXCEPTION;
```

٢ - إظهار الاستثناء في الجزء التنفيذي من الوحدة (Block)نتيجة أي حدث يُعتبر على أنه خطأ ويجب وقف التنفيذ بسببه باستخدام تعليمة RAISE متبوعة باسم الاستثناء .

```
Begin
….
RAISE Exception_name    ;
….
```

٣ - معالجة الاستثناء في جزء الاستثناءات في الوحدة (Block).:

تتم عملية المعالجة بمحاولة الإمساك بالاستثناء وذلك بكتابة اسم الاستثناء بعد كلمة WHEN ومن ثم كتابة الجمل المناسبة للمعالجة.

مثال : تقوم هذه الوحدة بتعديل الوصف لمنتج معين وذلك باستقبال الرقم والوصف الجديد للمنتج،وفي حالة عدم التعديل (رقم المنتج غير صحيح) يتم إظهار الاستثناء e_invalid_product الذي تم تعريفه .ومعالجة هذا الاستثناء في جزء الاستثناءات.

DECLARE		
EXCEPTION; e_invalid_product	→	تعريف الاستثناء
BEGIN product UPDATE descrip = '&product_description' SET prodid = &product_number; WHERE IF SQL%NOTFOUND THEN		
RAISE e_invalid_product;	→	إظهار الاستثناء نتيجة لعدم إتمام عملية التعديل
END IF; COMMIT; EXCEPTION		
WHEN e_invalid_product THEN DBMS_OUTPUT.PUT_LINE('Invalid product number.');	→	معالجة الاستثناء
END;		

- **استخدام SQLCODE و SQLERRM** :

- SQLCODE و SQLERRM عبارة عن دوال يمكن استخدامها في عملية معالجة الاستثناءات وذلك للاستفادة منها في عملية التعرف على الأخطاء التي تحدث خلال عملية التنفيذ وكذلك يمكن الاستفادة منها في تغيير الرسالة التي تظهر خلال عملية التنفيذ .
- SQLCODE: دالة تعيد رقم الخطأ الذي حدث .
- SQLERRM دالة تعيد نص الرسالة للخطأ الذي حدث.

SQLCODE	الوصف
٠	لم يظهر أي استثناء
١	استثناء معرف من قبل المستخدم
+١٠٠	NO_DATA_FOUND
رقم سالب	أي خطأ من أخطاء خادم ORACLE

ماذا يحدث عند عدم معالجة الاستثناء ؟

إن ظهور الاستثناء يؤدي إلى توقف الوحدة وإذا لم تتم معالجة هذا الاستثناء سينتقل إلى المكان الذي تمت منه عملية الاستدعاء لهذه الوحدة .وإذا لم تتم المعالجة في هذا المكان سينتقل إلى البيئه التي استدعته وهكذا .... حتى يصل أول مكان بدأت منه عملية التنفيذ .

```
DECLARE
 . . .
      EXCEPTION;        e_no_rows
EXCEPTION;      e_integrity
 PRAGMA EXCEPTION_INIT (e_integrity, -2292);
BEGIN
 FOR c_record IN emp_cursor LOOP
```

```
BEGIN
SELECT ...
UPDATE ...
IF SQL%NOTFOUND THEN
RAISE e_no_rows;
END IF;
EXCEPTION
WHEN e_integrity THEN ...
WHEN e_no_rows THEN ...
END;
```

يمكن معالجة الاستثناء في الوحدة الداخلية Sub Block ومن ثم الخروج إلى الوحدة الخارجية وتستمر عملية التنفيذ وإذا لم تتم عملية المعالجة فإن الاستثناء سينتقل إلى الوحدة الخارجية Main Block وبالتالي سيؤدي إلى وقف عملية التنفيذ ولكن يمكن عمل المعالجة قبل الخروج من Main Block

```
END LOOP;
EXCEPTION
 WHEN NO_DATA_FOUND THEN . . .
 WHEN TOO_MANY_ROWS THEN . . .
END;
```

يمكن معالجة الاستثناء في الوحدة الخارجية Main Block وإذا لم تتم عملية المعالجة فإن الاستثناء سينتقل إلى البيئة الخارجية في هذه الحالة إلى SQL *Plus

استخدام الإجراء RAISE_APPLICATION_ERROR :

وهو عبارة عن إجراء يتيح للمستخدم إظهار الأخطاء وتحديد نص الرسالة التي تعرض في حالة إظهار هذا الخطأ .لايستعمل هذا الإجراء إلا في الوحدات المخزنة في قاعدة البيانات مثل (Procedures, Functions).

RAISE_APPLICATION_ERROR(*error_number,message,[TRUE,FLASE]);*

error_number	رقم الخطأ وهو رقم يقوم المستخدم بتحديده. يسمح للمستخدم باستخدام الأرقام بين-٢٠٠٠٠ و -٢٠٩٩٩
message	أي رسالة يرغب المستخدم بإظهارها
[TRUE,FLASE]	اختيارية. وتعني TRUE وضع الخطأ فوق الخطأ السابق في ال Stack. وتعني FLASE الخطأ سيحل محل جميع الأخطاء السابقة .وهي الافتراضية (Defualt)

ويمكن استخدام RAISE_APPLICATION_ERROR في الجزء التنفيذي وكذلك في جزء الاستثناءات.

```
…
BEGIN
….

IF (v_grade  > 100 OR v_grade < 0) THEN
RAISE_APPLICATION_ERROR(-20210,'invlid grade  ');
END IF;
…..
END;
```

```
…
EXCEPTION
WHEN NO_DATA_FOUND THEN
RAISE_APPLICATION_ERROR(-20010,'invlid employee number');
…..
END;
```

تمارين

١ - قم بإنشاء وحدة (PL/SQL Block) لاسترجاع اسم الموظف حسب الراتب(إدخال الراتب) .

- إذا أعادت جملة الاسترجاع أكثر من صف يجب معالجة هذا الاستثناء وتخزين النص التالي More than one employee with a salary<*salary*> في جدول message .
- إذا لم تعد جملة الاسترجاع أي صف يجب معالجة هذا الاستثناء وتخزين النص التالي No employee with a salary<*salary*> في جدول message .
- إذا أعادت جملة الاسترجاع صفاً واحداً فقط قم بتخزين اسم الموظف والراتب في جدول message .
- قم بفحص الوحدة لأكثر من راتب .

```
RESULTS
------------------------------------------------------------
SMITH – 800
More than one employee with a salary of 3000
No employee with a salary of 6000
```

٢ - قم بعديل الملف *p8q3.sql* لإضافة معالج استثناءات لمعالجة الاستثناء الناتج عن إدخال رقم قسم غير موجود في جدول dept .

تأكد من عملية المعالجة وذلك بإدخال رقم قسم غير موجود .

```
Please enter the department number: 50
 Please enter the department location: HOUSTON

PL/SQL procedure successfully completed.

G_MESSAGE
--------------------------------------------------------------------------------
Department 50 is an invalid department
```

٣ - قم بإنشاء وحدة (PL/SQL Block) لطباعة عدد الموظفين الذين يزيد راتبهم أو يقل عن الراتب الُمدخل بمقدار ١٠٠ .

- إذا لم يوجد أي راتب ضمن هذا المجال قم بعمل استثناء وطباعة رسالة تخبر المستخدم بذلك .
- إذا كان هناك موظف أو أكثر ضمن هذا المجال يجب طباعة هذا العدد.
- إذا حدث أي خطأ آخر يجب معالجة هذا الاستثناء وطباعة الرسالة التالية (Some other error occurred) .

```
Please enter the salary: 800
PL/SQL procedure successfully completed.

G_MESSAGE
--------------------------------------------------------------------------------
There is/are 1 employee(s) with a salary between 700 and 900
```

```
Please enter the salary: 3000
PL/SQL procedure successfully completed.

G_MESSAGE
--------------------------------------------------------------------------------
There is/are 3 employee(s) with a salary between 2900 and 3100
```

```
Please enter the salary: 6000
PL/SQL procedure successfully completed.

G_MESSAGE
--------------------------------------------------------------------------------
There is no employee salary between 5900 and 6100
```



```
SQL> DESCRIBE emp
```

```
Name                                                    Null?    Type
 ------------------------------------------------------- -------- -------------
--
 EMPNO                                                   NOT NULL NUMBER(4)
 ENAME                                                            VARCHAR2(10)
 JOB                                                              VARCHAR2(9)
 MGR                                                              NUMBER(4)
 HIREDATE                                                         DATE
 SAL                                                              NUMBER(7,2)
 COMM                                                             NUMBER(7,2)
 DEPTNO                                                  NOT NULL NUMBER(2)
```

```
SQL> SELECT * FROM emp;
```

```
EMPNO ENAME      JOB              MGR HIREDATE        SAL       COMM     DEPTNO
------ ---------- --------- ---------- -------- ---------- ---------- ----------
  7839 KING       PRESIDENT            17/11/81       5000                    10
  7698 BLAKE      MANAGER         7839 01/05/81       2850                    30
  7782 CLARK      MANAGER         7839 09/06/81       2450                    10
  7566 JONES      MANAGER         7839 02/04/81       2975                    20
  7654 MARTIN     SALESMAN        7698 28/09/81       1250       1400         30
  7499 ALLEN      SALESMAN        7698 20/02/81       1600        300         30
  7844 TURNER     SALESMAN        7698 08/09/81       1500          0         30
  7900 JAMES      CLERK           7698 03/12/81        950                    30
  7521 WARD       SALESMAN        7698 22/02/81       1250        500         30
  7902 FORD       ANALYST         7566 03/12/81       3000                    20
  7369 SMITH      CLERK           7902 17/12/80        800                    20
  7788 SCOTT      ANALYST         7566 09/12/82       3000                    20
  7876 ADAMS      CLERK           7788 12/01/83       1100                    20
  7934 MILLER     CLERK           7782 23/01/82       1300                    10
```

```
SQL>  DESCRIBE dept
```

```
Name                                                    Null?    Type
 ------------------------------------------------------- -------- -------------
-----
 DEPTNO                                                  NOT NULL NUMBER(2)
 DNAME                                                            VARCHAR2(14)
 LOC                                                              VARCHAR2(13)
```

```
SQL> SELECT * FROM  dept;
```

```
    DEPTNO      DNAME              LOC
----------         --------------            -------------
        10          ACCOUNTING     NEW YORK
        20          RESEARCH          DALLAS
        30          SALES                  CHICAGO
        40         OPERATIONS      BOSTON
```

```
SQL>  DESCRIBE SALGRADE
```

```
Name                                                      Null?    Type
 ----------------------------------------------------- -------- -------------
-------
 GRADE                                                              NUMBER
 LOSAL                                                               NUMBER
 HISAL                                                                NUMBER
```

```
SQL> SELECT *  FROM SALGRADE;
```

```
     GRADE      LOSAL      HISAL
---------- -        ---------       ----------
         1                  700        1200
         2                 1201       1400
         3                 1401       2000
         4                 2001       3000
         5                 3001       9999
```

```
SQL>  DESCRIBE ord
```

```
Name                                                      Null?
Type
 -----------------------------------------------------              --------
 ORDID                                                     NOT NULL
NUMBER(4)
 ORDERDATE                                                                 DATE
 COMMPLAN
VARCHAR2(1)
 CUSTID                                                    NOT NULL    NUMBER(6)
 SHIPDATE
DATE
 TOTAL
NUMBER(8,2)
```

```
SQL>  SELECT *  FROM  ord;
```

```
   ORDID ORDERDAT C     CUSTID SHIPDATE      TOTAL
-------- -------- - ---------- -------- ----------
     610 07/01/87 A        101 08/01/87      101.4
     611 11/01/87 B        102 11/01/87         45
     612 15/01/87 C        104 20/01/87       5860
     601 01/05/86 A        106 30/05/86        2.4
     602 05/06/86 B        102 20/06/86         56
     604 15/06/86 A        106 30/06/86        698
     605 14/07/86 A        106 30/07/86       8324
     606 14/07/86 A        100 30/07/86        3.4
     609 01/08/86 B        100 15/08/86       97.5
     607 18/07/86 C        104 18/07/86        5.6
     608 25/07/86 C        104 25/07/86       35.2
     603 05/06/86          102 05/06/86        224
     620 12/03/87          100 12/03/87       4450
     613 01/02/87          108 01/02/87       6400
```

```
     614 01/02/87         102 05/02/87      23940
     616 03/02/87         103 10/02/87        764
     619 22/02/87         104 04/02/87       1260
     617 05/02/87         105 03/03/87      46370
     615 01/02/87         107 06/02/87        710
     618 15/02/87 A       102 06/03/87     3510.5
     621 15/03/87 A       100 01/01/87        730
```

SQL> DESCRIBE PRODUCT

```
Name                                                      Null?    Type
 ----------------------------------------------------- -------- -------------
----------
 PRODID                                                   NOT NULL NUMBER(6)
 DESCRIP                                                            VARCHAR2(30)
```

```
SQL>  SELECT *  FROM  PRODUCT ;
```

```
PRODID                 DESCRIP
----------                ------------------------------
    100860             ACE TENNIS RACKET I
    100861             ACE TENNIS RACKET II
    100870             ACE TENNIS BALLS-3 PACK
    100871             ACE TENNIS BALLS-6 PACK
    100890             ACE TENNIS NET
    101860             SP TENNIS RACKET
    101863             SP JUNIOR RACKET
    102130             RH: "GUIDE TO TENNIS"
    200376             SB ENERGY BAR-6 PACK
    200380             SB VITA SNACK-6 PACK
```

SQL> DESCRIBE ITEM

```
Name                           Null?            Type
 -------------------------    --------        ------------
 ORDID                          NOT NULL         NUMBER(4)
 ITEMID                         NOT NULL         NUMBER(4)
 PRODID                                          NUMBER(6)
 ACTUALPRICE                                     NUMBER(8,2)
 QTY                                             NUMBER(8)
 ITEMTOT                                         NUMBER(8,2)
```

SQL> SELECT * FROM ITEM;

```
  ORDID     ITEMID     PRODID ACTUALPRICE        QTY    ITEMTOT
------- ---------- ---------- ----------- ---------- ----------
    610          3     100890          58          1         58
    611          1     100861          45          1         45
    612          1     100860          30        100       3000
    601          1     200376         2.4          1        2.4
    602          1     100870         2.8         20         56
```

604	1	100890	58	3	174
604	2	100861	42	2	84
604	3	100860	44	10	440
603	2	100860	56	4	224
610	1	100860	35	1	35
610	2	100870	2.8	3	8.4
613	4	200376	2.2	200	440
614	1	100860	35	444	15540
614	2	100870	2.8	1000	2800
612	2	100861	40.5	20	810
612	3	101863	10	150	1500
620	1	100860	35	10	350
620	2	200376	2.4	1000	2400
620	3	102130	3.4	500	1700
613	1	100871	5.6	100	560
613	2	101860	24	200	4800
613	3	200380	4	150	600
619	3	102130	3.4	100	340
617	1	100860	35	50	1750
617	2	100861	45	100	4500
614	3	100871	5.6	1000	5600
616	1	100861	45	10	450
616	2	100870	2.8	50	140
616	3	100890	58	2	116
616	4	102130	3.4	10	34
616	5	200376	2.4	10	24
619	1	200380	4	100	400
619	2	200376	2.4	100	240
615	1	100861	45	4	180
607	1	100871	5.6	1	5.6
615	2	100870	2.8	100	280
617	3	100870	2.8	500	1400
617	4	100871	5.6	500	2800
617	5	100890	58	500	29000
617	6	101860	24	100	2400
617	7	101863	12.5	200	2500
617	8	102130	3.4	100	340
617	9	200376	2.4	200	480
617	10	200380	4	300	1200
609	2	100870	2.5	5	12.5
609	3	100890	50	1	50
618	1	100860	35	23	805
618	2	100861	45.11	50	2255.5
618	3	100870	45	10	450
621	1	100861	45	10	450
621	2	100870	2.8	100	280
615	3	100871	5	50	250
608	1	101860	24	1	24
608	2	100871	5.6	2	11.2
609	1	100861	35	1	35
606	1	102130	3.4	1	3.4
605	1	100861	45	100	4500
605	2	100870	2.8	500	1400
605	3	100890	58	5	290
605	4	101860	24	50	1200

```
     605          5      101863             9          100          900
     605          6      102130           3.4           10           34
     612          4      100871           5.5          100          550
     619          4      100871           5.6           50          280
```

SQL> DESCRIBE PRICE

```
Name                                                   Null?    Type
 ------------------------------------------------------- -------- -------------
------
 PRODID                                                  NOT NULL NUMBER(6)
 STDPRICE                                                         NUMBER(8,2)
 MINPRICE                                                         NUMBER(8,2)
 STARTDATE                                                        DATE
 ENDDATE                                                          DATE
```

SQL> SELECT * FROM PRICE;

```
    PRODID   STDPRICE   MINPRICE STARTDAT ENDDATE
---------- ---------- ---------- -------- --------
    100871        4.8        3.2 01/01/85 01/12/85
    100890         58       46.4 01/01/85
    100860         35         28 01/06/86
    100860         30         24 01/01/85 31/12/85
    100861         45         36 01/06/86
    100861         39       31.2 01/01/85 31/12/85
    100870        2.8        2.4 01/01/86
    100870        2.4        1.9 01/01/85 01/12/85
    100871        5.6        4.8 01/01/86
    101860         24         18 15/02/85
    101863       12.5        9.4 15/02/85
    102130        3.4        2.8 18/08/85
    200376        2.4       1.75 15/11/86
    200380          4        3.2 15/11/86
```

SQL> DESCRIBE CUSTOMER

```
Name                                                   Null?    Type
 ------------------------------------------------------- -------- -------------
--
 CUSTID                                                  NOT NULL NUMBER(6)
 NAME                                                             VARCHAR2(45)
 ADDRESS                                                          VARCHAR2(40)
 CITY                                                             VARCHAR2(30)
 STATE                                                            VARCHAR2(2)
 ZIP                                                              VARCHAR2(9)
 AREA                                                             NUMBER(3)
 PHONE                                                            VARCHAR2(9)
 REPID                                                   NOT NULL NUMBER(4)
 CREDITLIMIT                                                      NUMBER(9,2)
 COMMENTS                                                         LONG
```

CUSTID	NAME	ADDRESS
100	JOCKSPORTS	345 VIEWRIDGE
101	TKB SPORT SHOP	490 BOLI RD.
102	VOLLYRITE	9722 HAMILTON
103	JUST TENNIS	HILLVIEW MALL
104	EVERY MOUNTAIN	574 SUYYYYY RD.
105	K + T SPORTS	3476 EL PASEO
106	SHAPE UP	908 SEQUOIA
107	WOMENS SPORTS	VALCO VILLAGE
108	NORTH WOODS HEALTH AND FITNESS SUPPLY CENTER	98 LONE PINE WAY

CITY	ST	ZIP	AREA	PHONE	REPID	CREDITLIMIT
BELMONT	CA	96711	415	598-6609	7844	5000
REDWOOD CITY	CA	94061	415	368-1223	7521	10000
BURLINGAME	CA	95133	415	644-3341	7654	7000
BURLINGAME	CA	97544	415	677-9312	7521	3000
CUPERTINO	CA	93301	408	996-2323	7499	10000
SANTA CLARA	CA	91003	408	376-9966	7844	5000
PALO ALTO	CA	94301	415	364-9777	7521	6000
SUNNYVALE	CA	93301	408	967-4398	7499	10000
HIBBING	MN	55649	612	566-9123	7844	8000

CUSTID	COMMENTS
100	Very friendly people to work with -- sales rep likes to be called Mike.
101	Rep called 5/8 about change in order - contact shipping.
102	Company doing heavy promotion beginning 10/89. Prepare for large orders during
103	Contact rep about new line of tennis rackets.
104	Customer with high market share (23%) due to aggressive advertising.
105	Tends to order large amounts of merchandise at once. Accounting is considering
106	Support intensive. Orders small amounts (< 800) of merchandise at a time.
107	First sporting goods store geared exclusively towards women. Unusual Promotional style and very willing to take chances towards new products!
108	

المحتويات

تقدر المؤسسة العامة للتعليم الفني والتدريب المهني الدعم

المالي المقدم من شركة بي آيه إي سيستمز (العمليات) المحدودة

GOTEVOT appreciates the financial support provided by BAE SYSTEMS

BAE SYSTEMS